Titre: Title:	Simulations stochastiques par rapiéçage de motifs avec contrôle des statistiques
Auteur: Author:	Corentin Faucher
Date:	2013
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Faucher, C. (2013). Simulations stochastiques par rapiéçage de motifs avec contrôle des statistiques [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1087/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1087/
Directeurs de recherche: Advisors:	Antoine Saucier, & Denis Marcotte
Programme: Program:	Mathématiques de l'ingénieur

UNIVERSITÉ DE MONTRÉAL

SIMULATIONS STOCHASTIQUES PAR RAPIÉÇAGE DE MOTIFS AVEC CONTRÔLE DES STATISTIQUES

CORENTIN FAUCHER
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AVRIL 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

SIMULATIONS STOCHASTIQUES PAR RAPIÉÇAGE DE MOTIFS
AVEC CONTRÔLE DES STATISTIQUES

présentée par : FAUCHER Corentin
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. BOYER François-Raymond, Ph.D., président
M. SAUCIER Antoine, Ph.D., membre et directeur de recherche
M. MARCOTTE Denis, Ph.D., membre et codirecteur de recherche
M. GALINIER Philippe, Ph.D., membre
M. RENARD Philippe, Doctorat, membre

À Charlie

REMERCIEMENTS

Je souhaiterais tout d'abord remercier mon directeur de recherche, M. Antoine Saucier, pour son appui et ses précieux conseils dans la réalisation de ce travail. Grâce à lui, j'ai adopté une méthodologie de travail rigoureuse qui m'a permis de structurer mes idées. Je veux le remercier aussi pour la patience dont il a fait preuve à mon égard.

Je souhaite également remercier mon co-directeur de recherche, M. Denis Marcotte, pour ses bons conseils et commentaires. Il m'a fait découvrir le domaine de la géostatistique et m'a référé à différents sujets essentiels pour la réalisation de mon travail.

Je tiens beaucoup à remercier le président, M. François-Raymond Boyer, et les membres du jury, M. Philippe Galinier et M. Philippe Renard, pour leur participation à l'examen de cette thèse.

Je remercie le Conseil de recherches en sciences naturelles et en génie du Canada pour l'attribution du financement du programme de subventions à la découverte.

Je remercie mes amis et collègues de travail (dont Roland pour ses sages suggestions ;)). Finalement, je voudrais remercier les membres de ma famille et en particulier ma fiancée de leur soutien.

RÉSUMÉ

Dans cette thèse, nous présentons deux méthodes de simulation stochastique qui procèdent de manière similaire à la construction d'un casse-tête en assemblant des morceaux carrés d'image, d'où le nom de *simulations stochastiques par rapiéçage de motifs*. Les morceaux sont prélevés dans une *image de référence* qui fournit les informations sur les propriétés statistiques du champ aléatoire à reproduire. Nos méthodes de simulation mettent l'accent sur la continuité des motifs lors du rapiéçage. De plus, nos méthodes de simulation permettent de respecter la présence de *données conditionnantes* (*hard data*). Pour ce conditionnement, nous limitons le choix des morceaux d'image à ceux qui respectent les données conditionnantes.

Afin d'éviter le problème de dérive des statistiques, nous présentons une méthode de contrôle reposant sur le contrôle de l'histogramme de la moyenne locale. Pour contrôler cet histogramme, nous classons dans une partition les morceaux de l'image de référence en fonction de la valeur moyenne de leurs pixels. En plus de pouvoir reproduire l'histogramme de la moyenne locale de l'image de référence, les simulations peuvent reproduire un histogramme quelconque défini par l'usager.

La première méthode de simulation, appelée *patchwork simulation method* (PSM), procède de gauche à droite et de haut en bas en collant des morceaux d'image de l'image de référence. Cette méthode de simulation est apparentée à un champ de Markov où le contrôle de la moyenne locale s'effectue en choisissant adéquatement les probabilités de transition.

La deuxième méthode de simulation, appelée *corrective pattern-matching simulation* (CPMS), procède en effectuant des corrections de manière itérative. Cette méthode est apparentée à un algorithme glouton, car à chaque étape elle apporte une correction à un emplacement présentant une grande erreur dans l'objectif de tendre vers un optimum global.

Pour différents types d'images de synthèse et pour une image d'un mélange de polymère, nous montrons que nos simulations respectent les données conditionnantes, reproduisent l'aspect visuel des images de référence et permettent d'obtenir des histogrammes de la moyenne locale qui sont statistiquement conforme à l'histogramme visé.

ABSTRACT

In this thesis, we present two stochastic simulation methods which build two-dimensional images by assembling together square image pieces like a jigsaw puzzle. The image pieces are taken from a reference image that provides the statistical information on the simulated field. Our methods emphasize pattern continuity in the simulated image. This is achieved by suitably selecting the image pieces to fit to the existing data. Our simulations are also constrained to respect conditional data called *hard data*. This conditioning is achieved by limiting the image pieces drawn from the reference image to the pieces that honor the hard data.

To avoid statistical drift in our simulation methods, we introduce different ways to control the local-mean histogram. The local-mean is the average value of the pixels in a square image piece. The statistical control is achieved by partitioning all the possible image pieces of the reference image according to their local-mean. The simulations can then reproduce the local-mean histogram of the reference image. Furthermore, the simulations can also reproduce a user-defined local-mean histogram and thus produce original random fields.

The first simulation method, called *patchwork simulation method* (PSM), proceeds unilaterally by sticking image pieces from left to right and from top to bottom. In the PSM, the local-mean histogram control is achieved by suitably adjusting the transition probabilities that associate the added data to an existing neighborhood in the partly simulated image.

The second simulation method, called *corrective pattern-matching simulation* (CPMS), proceeds iteratively by making local corrections to the simulated image. The CPMS proceeds as a greedy algorithm in the sense that, at each step, the simulation corrects a location that presents a large error in the hope to tend towards a global optimum.

For several types of synthetic images and one polymer blend image, we show that our simulations respect the conditioning data, reproduce faithfully the visual appearance of the reference images and are statistically compatible with the target local-mean histograms.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

Symbole	Définition
$\boldsymbol{a}(\boldsymbol{s})$	État associé au carré de taille $3L/4$ dans le coin inférieur droit de l'état $\boldsymbol{s}$.
$A(\boldsymbol{u})$	Ensemble de tous les états $\boldsymbol{s} \in \mathcal{S}_r$ tels que $\boldsymbol{u}(\boldsymbol{s})$ est le plus proche de $\boldsymbol{u}$.
$A_n(\boldsymbol{u})$	Ensemble de tous les états $\boldsymbol{s} \in \mathcal{S}_r$ qui respectent les données conditionnantes locales à l'étape n tels que $\boldsymbol{u}(\boldsymbol{s})$ est le plus proche de $\boldsymbol{u}$.
$A_n^*(\boldsymbol{u})$	Ensemble de tous les états $\boldsymbol{s} \in \mathcal{S}_r$ qui respectent les données conditionnantes locales à l'étape n tels que $\boldsymbol{u}(\boldsymbol{s}^*)$ est le plus proche de $\boldsymbol{u}$.
$A^\dagger(\boldsymbol{u})$	Ensemble de tous les états $\boldsymbol{s} \in \mathcal{S}_r$ tels que $\boldsymbol{u}(\boldsymbol{s}) = \boldsymbol{u}$.
$A'(\boldsymbol{u})$	Ensemble de tous les états $\boldsymbol{s} \in \mathcal{S}_r$ tels que $\boldsymbol{u}(\boldsymbol{s})$ est le plus proche de $\boldsymbol{u}$ en utilisant une métrique pondérée.
B_j	Région carrée de taille L appelée *boîte* contenue dans le support d'une image et indexée par l'indice $j \in \{1, \ldots, m\}$.
B_j^*	Région carrée de taille $2L$ centrée au même endroit que la boîte B_j.
$\mathcal{B}_r$ et $\mathcal{B}_s$	L'ensemble de toutes les boîtes de taille L contenues dans l'image de référence et l'image simulée respectivement.
$\mathcal{B}_r^*$ et $\mathcal{B}_s^*$	L'ensemble de toutes les boîtes de taille $2L$ contenues dans l'image de référence et l'image simulée respectivement.
$c(\boldsymbol{s})$	Nombre d'occurrences de l'état $\boldsymbol{s}$ dans l'image de référence.
C_i	$i^{\text{ième}}$ classe de la moyenne locale, $i \in \{1, 2, ..., M\}$, définie par $C_i = \{\boldsymbol{s} \in \mathcal{S}_r : \varphi_{i-1} \leq \phi(\boldsymbol{v}(\boldsymbol{s})) < \varphi_i\}$ (PSM) ou par $C_i = \{\boldsymbol{u} \in \mathcal{U}_r : \varphi_{i-1} \leq \phi(\boldsymbol{s}(\boldsymbol{u})) < \varphi_i\}$ (CPMS).
$C_i(\boldsymbol{z}_j)$	Sous-ensemble de C_i ne contenant que les états qui respectent les données conditionnantes $\boldsymbol{z}_j$.
$D^2(\boldsymbol{x})$	Statistique du test du χ^2 utilisée pour vérifier si une image simulée est statistiquement compatible avec l'histogramme visé.

$d_{\boldsymbol{y}}(\boldsymbol{y}_i, \boldsymbol{y}_j)$	Distance entre deux états $\boldsymbol{y}_i$ et $\boldsymbol{y}_j$. Ici, le symbole $\boldsymbol{y} \in \{\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{z}\}$ dénote le type d'état.
$e(j)$ et $E(\boldsymbol{x})$	L'erreur locale dans la boîte $B_j \in \Omega$ et l'erreur globale dans l'image simulée $\boldsymbol{x}$.
$e_{\text{hd}}(j)$ et $E_{\text{hd}}(\boldsymbol{x})$	L'erreur locale sur les données conditionnantes dans la boîte $B_j \in \Omega$ et l'erreur globale sur les données conditionnantes dans l'image simulée $\boldsymbol{x}$.
$e_{\text{pc}}(j)$ et $E_{\text{pc}}(\boldsymbol{x})$	L'erreur locale sur la conformité aux motifs (**p**attern **c**onformity) dans la boîte $B_j \in \Omega$ et l'erreur globale, dans l'image simulée $\boldsymbol{x}$, sur la conformité aux motifs.
$e_{\text{stat}}(j)$ et $E_{\text{stat}}(\boldsymbol{x})$	L'erreur locale statistique dans la boîte $B_j \in \Omega$ et l'erreur globale statistique dans l'image simulée $\boldsymbol{x}$.
$\boldsymbol{f}$	Vecteur $\boldsymbol{f} \in \mathbb{R}^M$ contenant les fréquences relatives des éléments de chaque classe de l'histogramme de la moyenne locale pour une image simulée.
$i(\boldsymbol{s})$	Indice de classe de la moyenne locale pour l'état $\boldsymbol{s}$ tel que $\boldsymbol{s} \in C_{i(\boldsymbol{s})}$.
L	La taille de la boîte de l'état $\boldsymbol{s}$, correspond à la taille d'une boîte B_j.
M	Nombre $M \in \{2, 3, ...\}$ de classes pour l'histogramme de la moyenne locale.
n	Indice numérotant les étapes d'une simulation.
N	Nombre d'éléments dans la grille de simulation (correspond au nombre total d'étapes pour la simulation unilatérale).
$\boldsymbol{p}$	Vecteur $\boldsymbol{p} \in \mathbb{R}^M$ contenant les probabilités $p(i)$ d'observer dans l'image simulée un état $\boldsymbol{s}$ dans la $i^{\text{ième}}$ classe de la moyenne locale.
$\boldsymbol{p}_r$	Vecteur $\boldsymbol{p}_r \in \mathbb{R}^M$ contenant les probabilités relatives à chacune des classes de la moyenne locale pour l'image de référence.
$\boldsymbol{p}_t$	Vecteur $\boldsymbol{p}_t \in \mathbb{R}^M$ contenant les probabilités désirées de chacune des classes de la moyenne locale.

$p_{j,k}$	Probabilités de transition définies par $p_{j,k} := P\left[\boldsymbol{S}(n) = \boldsymbol{s}_k | \boldsymbol{U}(n) = \boldsymbol{u}_j\right]$. La probabilité $p_{j,k}$ correspond à la probabilité de coller l'état $\boldsymbol{s}_k$, étant donné le voisinage $\boldsymbol{u}_j$.
$\boldsymbol{s}$	État utilisé pour ajouter des données à la simulation. C'est un vecteur contenant les valeurs des pixels contenus dans une boîte carrée de taille L.
$\boldsymbol{s}^*$	État $\boldsymbol{s}$ étendu utilisé pour les simulations unilatérales avec des données conditionnantes.
$\boldsymbol{s}(\boldsymbol{u})$	Dans la simulation CPMS : état $\boldsymbol{s}$ associé à la partie centrale de l'état $\boldsymbol{u}$.
$\boldsymbol{S}$	Vecteur aléatoire associé à un état $\boldsymbol{s}$ choisi au hasard dans une image simulée.
$\boldsymbol{S}_r$	Vecteur aléatoire associé à un état $\boldsymbol{s}$ choisi au hasard dans l'image de référence de façon équiprobable.
$\boldsymbol{S}_t$	Vecteur aléatoire associé à un état $\boldsymbol{s}$ ayant la distribution de probabalité désirée.
$\mathcal{S}_r$	Ensemble de tous les états distincts $\boldsymbol{s}$ observés dans l'image de référence.
$\mathcal{S}_{r,\,n}$	Sous-ensemble de $\mathcal{S}_r$ contenant les états qui respectent les données conditionnantes locales à l'étape n.
$\mathcal{S}_r^*$	Ensemble de tous les états étendus distincts $\boldsymbol{s}^*$ observés dans l'image de référence.
$\mathcal{S}_{r,\,n}^*$	Sous-ensemble de $\mathcal{S}_r^*$ contenant les états qui respectent les données conditionnantes locales à l'étape n.
$\boldsymbol{u}$	État utilisé pour considérer le voisinage, appelé aussi état de départ. Il est en forme de L pour la simulation unilatérale et en forme de carré de taille $2L$ pour la simulation CPMS.
$\boldsymbol{u}(\boldsymbol{s})$	Dans la simulation unilatérale : état $\boldsymbol{u}$ associé à la partie en forme de L d'un état $\boldsymbol{s}$.

$\boldsymbol{U}(n)$	Dans la simulation unilatérale : vecteur aléatoire associé à l'état $\boldsymbol{u}$ observé à la $n^{\text{ième}}$ étape de la simulation.
$\mathcal{U}_r$	Ensemble de tous les états $\boldsymbol{u}$ distincts observés dans l'image de référence.
$\mathcal{U}_r(\boldsymbol{z}_j)$	Sous-ensemble des états de $\mathcal{U}_r$ qui respectent les données conditionnantes locales $\boldsymbol{z}_j$.
$\boldsymbol{v}(\boldsymbol{s})$	Dans la simulation unilatérale : quartier de l'état $\boldsymbol{s}$ associé au carré de taille $L/2$ dans le coin inférieur droit de l'état $\boldsymbol{s}$.
$\boldsymbol{V}_n$	Vecteur aléatoire associé au quartier $\boldsymbol{v}(\boldsymbol{S}(n))$, une fois la simulation terminée.
$w_{\boldsymbol{y}}(p)$	$p^{\text{ième}}$ poids utilisé pour pondérer les valeurs des pixels dans la distance $d_{\boldsymbol{y}}$.
$\boldsymbol{x}$	Vecteur contenant toutes les valeurs des pixels de l'image simulée.
$\boldsymbol{z}_j$	Vecteur contenant les valeurs des données conditionnantes retrouvées dans la boîte B_j.
$\boldsymbol{z}(\boldsymbol{s}_j)$	Vecteur contenant les composantes de $\boldsymbol{s}_j$ correspondantes aux emplacements des données conditionnantes $\boldsymbol{z}_j$.
Z ou $\boldsymbol{Z}$	Variable ou vecteur aléatoire du champ aléatoire étudié.
α	Constante de normalisation.
$\beta(M)$	Coefficient fonction du nombre de classes M utilisées pour le contrôle de la moyenne locale dans la méthode CPMS.
γ	Coefficient utilisé pour pondérer la contribution des erreurs relatives $\varepsilon(i)$ dans le choix d'un état $\boldsymbol{u}_k$.
$\delta_j(i)$	Distance entre l'état $\boldsymbol{u}_j$ et la classe $C_i(\boldsymbol{z}_j)$.
Δ_j	Plage des valeurs $\delta_j(i)$ avec $i \in \{1, \ldots, M\}$, c'est-à-dire $\Delta_j = \max_i \delta_j(i) - \min_i \delta_j(i)$.
Δ	Valeur médiane des Δ_j pour les N dernières étapes de la simulation.
$\varepsilon(i)$	Erreur relative pour la $i^{\text{ième}}$ classe de la moyenne locale, plus précisément $\varepsilon(i) = f(i) - p_t(i)$.

μ_1 et μ_2	Coefficients de pénalité utilisés pour pondérer l'erreur sur les données conditionnantes $E_{\text{hd}}(\boldsymbol{x})$ et l'erreur statistique $E_{\text{stat}}(\boldsymbol{x})$ respectivement.
σ_i	Écart type des erreurs relatives $\varepsilon(i)$ pour une image à la limite de l'acceptabilité statistique selon le test du chi-deux.
$\bar{\sigma}$	Moyenne pondérée des σ_i.
$\phi(\boldsymbol{y})$	Moyenne locale : valeur moyenne des pixels de l'état $\boldsymbol{y}$ ($\boldsymbol{y}$ est de type $\boldsymbol{v}$ ou $\boldsymbol{s}$ dans les simulations PSM ou CPMS respectivement).
φ_i	Valeurs de la moyenne locale utilisées pour délimiter les classes C_i.
χ^2	Variable aléatoire suivant une loi du chi-deux.
$\omega(i)$	Poids $\omega(i) > 0$, $i \in \{1, 2, ..., M\}$ utilisés pour pondérer les différentes classes C_i (dans la méthode PSM).
$\boldsymbol{\omega}^*$	Vecteur des poids $\omega(i)$ optimaux pour obtenir des simulations respectant l'histogramme de la moyenne locale.
Ω	Recouvrement de boîtes B_j qui recouvrent toute la surface de l'image simulée à l'exception de bandes de largeur $L/2$ sur les bords de l'image.
$\stackrel{\text{d}}{=}$	Égalité en distribution (utilisé pour les variables aléatoires).

INTRODUCTION

Lorsqu'une partie d'une peinture nous est cachée, il est inévitable que nous imaginions la partie manquante. De même, sans avoir à y réfléchir, nous pouvons concevoir dans leur entièreté la plupart des objets que nous voyons tous les jours et qui apparaissent partiellement dans notre champ de vision. Dans les domaines minier et géologique, il est utile de concevoir ce que pourraient être les distributions de différentes ressources. Il peut s'agir, par exemple, de réservoirs sous-terrains ou de gisements de minerais. Bien entendu, ces minerais ne peuvent pas être observés à l'œil nu et nous devons nous fier à des mesures (les mesures sismiques, par exemple). Grâce à ces mesures, il peut ensuite être possible d'imaginer quelles seraient les distributions spatiales des différents minerais du sous-sol afin d'en faire la cartographie la plus réaliste possible. Ces prédictions, faites sur les concentrations de minerais, sont d'une grande importance, car elles permettent d'estimer les revenus potentiels d'un gisement.

L'ensemble des techniques permettant d'estimer les distributions spatiales des ressources géologiques forment la géostatistique. Afin de décrire la concentration des minerais en fonction de leur position, nous utilisons des méthodes de simulation géostatistique. Les simulations doivent tenir compte, autant que possible, de toutes les connaissances à priori du milieu étudié. Ces connaissances peuvent être des statistiques telles que les concentrations moyennes des minerais, les données recueillies à partir d'échantillons prélevés à des emplacements précis, les mesures effectuées sur la perméabilité du milieu, etc.

Une première catégorie de simulations géostatistiques est celle des simulations qui permettent de reproduire les statistiques dites deux-points. Dans ce cas, les valeurs simulées respectent une certaine valeur moyenne et la covariance entre deux valeurs simulées respecte la covariance attendue. Pour ce type de simulations, il suffit de connaître la valeur moyenne et les covariances.

Une deuxième catégorie de simulations géostatistiques est celle des simulations qui reproduisent les statistiques dites multi-points. Ces statistiques permettent de décrire des structures plus complexes en quantifiant les dépendances mutuelles entre plus de deux points de l'espace. Afin de décrire ces statistiques multi-points, nous utilisons une *image de référence*.

Il s'agit d'une image que l'on estime être représentative des structures réelles du milieu étudié. Grâce à cette image de référence, une méthode de simulation multi-point peut générer des images qui tiennent compte de l'information jugée pertinente pour représenter le mieux possible les structures du milieu étudié.

Dans cette thèse, nous présentons des méthodes de construction d'images fonctionnant de manière similaire à la construction d'un casse-tête. Tout d'abord, l'image de référence est découpée en morceaux carrés de taille $L \times L$. Ensuite, comme il est illustré à la figure 0.1, ces morceaux sont placés dans M ensembles en fonction de leur valeur moyenne. Ces ensembles sont appelés des *classes*. En ayant préclassé les morceaux de la sorte, nous pouvons privilégier le choix de la pige dans une classe plutôt que dans une autre. De cette façon, non seulement nous pouvons ajuster la proportion des pixels noirs et blancs, mais nous pouvons également nous assurer qu'une grande variété de morceaux d'image soit représentée. À l'inverse, nous pouvons décider d'éviter le choix de morceaux provenant de certaines classes et ainsi réaliser des casse-têtes complètement différents de l'image de référence.

Si aucune classe n'est privilégiée par rapport aux autres, nous choisissons simplement le morceau qui s'agence le mieux sans nous soucier de sa classe d'appartenance. Dans ce cas, certaines classes pourraient se trouver naturellement délaissées si leurs morceaux n'ont pas tendance à bien s'agencer avec les autres. Nous appelons ce phénomène la *dérive statistique*. Ce phénomène dépend de la méthode de construction employée. Nos méthodes de construction permettent d'éviter cette dérive en favorisant certaines classes, dans le but de garder le contrôle sur la distribution des morceaux provenant de chacune des classes. Nous appelons ce contrôle le *contrôle de la moyenne locale*.

Objectifs de la thèse

Dans cette thèse, l'objectif principal est de développer de nouvelles méthodes de simulation multi-point qui permettent de reproduire les propriétés d'une image de référence et de contrôler les statistiques de tout ordre. Ainsi, les images simulées reproduiront la texture et la structure de l'image de référence. Par exemple, si notre but est de simuler un réseau de chenaux tel qu'à la figure 1.2, les réalisations devront être en mesure de produire un réseau de chenaux interconnectés et être capable de reproduire des statistiques telles que la moyenne et

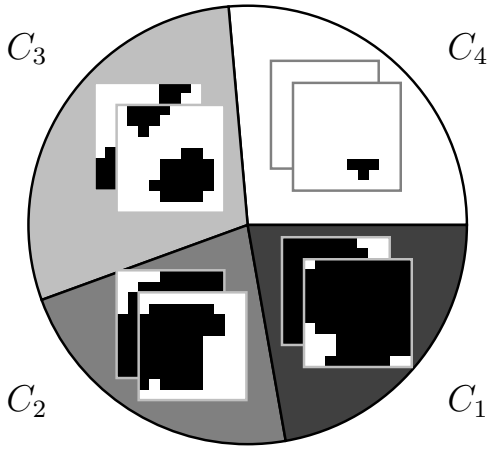


Figure 0.1 Partitions des morceaux de l'image de référence. Les morceaux sont des carrés de côté $L = 10$ pixels. Les morceaux sont classés dans $M = 4$ classes en fonction de la valeur moyenne de leurs pixels (noir et blanc correspondent respectivement à 0 et 1).

la covariance. De plus, nous voulons être en mesure de contrôler la distribution de la moyenne locale ou, plus précisément, l'histogramme de la moyenne locale. Ce contrôle nous permet d'adapter les images simulées afin d'approcher une distribution voulue. Plus précisément, nos objectifs sont de :

1. développer une méthode unidirectionnelle permettant d'assurer une bonne continuité dans l'image tout en contrôlant l'histogramme de la moyenne locale ;
2. développer une méthode de simulation séquentielle itérative de type *algorithme glouton* permettant d'assurer une bonne continuité dans l'image, tout en contrôlant l'histogramme de la moyenne locale ;
3. adapter cette dernière méthode séquentielle à une approche multi-échelle permettant de mieux reproduire les grandes structures tout en contrôlant les statistiques de l'image simulée ;
4. adapter les méthodes proposées pour permettre l'inclusion de données conditionnantes ;
5. adapter les méthodes proposées pour permettre le contrôle de l'histogramme de la moyenne locale.

Contributions originales

Les principales contributions apportées dans cette thèse sont d'avoir :

1. proposé une nouvelle métrique permettant de coller des morceaux d'image qui s'agencent le mieux possible avec leur voisinage ;
2. proposé une méthode de simulation par rapiéçage unilatérale
   i. s'adaptant à des données conditionnantes ;
   ii. utilisant deux méthodes de contrôle des statistiques pour corriger le problème de dérive observé fréquemment dans les simulations multi-points ;
3. proposé une méthode de simulation par rapiéçage itérative
   i. utilisant une méthode de quantification de l'erreur locale pour détecter les emplacements à corriger ;
   ii. effectuant des corrections locales en tenant compte de trois objectifs globaux : la conformité des motifs avec ceux de l'image de référence, la conformité aux données conditionnantes, la conformité à l'histogramme de la moyenne locale.

Plan de la thèse

Le chapitre 1 résume les méthodes de simulation séquentielle qui ont contribué à l'élaboration de nos méthodes de simulation par rapiéçage de motifs.

Le chapitre 2 donne une vue d'ensemble de nos méthodes de simulation par rapiéçage de motifs.

Le chapitre 3 présente le choix de la métrique qui permet de choisir à chaque étape d'une simulation le morceau qui s'agence le mieux avec son voisinage.

Le chapitre 4 décrit notre première méthode de simulation par rapiéçage de motifs appelée *patchwork simulation method* (PSM). Cette méthode est unilatérale, c'est-à-dire qu'elle procède de gauche à droite et de haut en bas en collant des morceaux de l'image de référence.

Le chapitre 5 décrit la deuxième méthode de simulation par rapiéçage de motifs appelée *corrective pattern-matching simulation* (CPMS). Cette méthode procède de manière itérative en effectuant des corrections locales.

Le chapitre 6 propose une approche multi-échelle pouvant être appliquée à la méthode CPMS.

Au chapitre 7, nous discutons de nos méthodes de simulation.

CHAPITRE 1

REVUE DE LITTÉRATURE

1.1 Les simulations point par point

Pour modéliser un environnement particulier, nous utilisons la notion de *champ aléatoire*. Un champ aléatoire est l'extension en deux ou trois dimensions d'un processus aléatoire. Une image constituée de N pixels de valeur $Z(n)$, $n = 1, \ldots, N$, définit le champ aléatoire $\mathcal{Z} = \{Z(1), Z(2), \ldots, Z(N)\}$. Par exemple, un champ aléatoire est dit gaussien et stationnaire s'il peut être entièrement caractérisé par sa moyenne $\mathrm{E}\,[Z(n)] = \mu$, sa variance $\mathrm{Var}\,[Z(n)] = \sigma^2$ et sa covariance $C(\boldsymbol{h}) = \mathrm{Cov}\,[Z(n), Z(m)]$, où $\boldsymbol{h}$ est le vecteur séparant les deux points d'indices n et m. Autrement dit, un champ aléatoire gaussien est complètement défini par ses deux premiers moments.

Dans cette thèse, nous considérons des méthodes de simulation séquentielle produisant des *images simulées*. Une simulation est habituellement conditionnée par la présence de *données conditionnantes* ou, en anglais, *hard data*. La valeur d'une donnée conditionnante est dénotée par la variable Z. Initialement, seules les données conditionnantes sont connues. Les valeurs des autres pixels sont indéterminées. Dans une simulation séquentielle *point par point*, les pixels de l'image simulée sont visités à tour de rôle et une valeur leur est attribuée lors de leur visite. La valeur attribuée est conditionnelle aux données conditionnantes et aux valeurs des pixels déjà simulés. La valeur d'un pixel attribuée par la simulation est dénotée par la variable S. Il est important de garder à l'esprit qu'une simulation ne permet pas d'obtenir l'image réelle qui est cachée (le champ aléatoire $\mathcal{Z}$), mais permet plutôt d'obtenir une réalisation possible de ce que pourrait être l'image cachée.

À la figure 1.1, nous illustrons une simulation séquentielle en cours. Les pixels ne peuvent prendre que les valeurs 0 ou 1 (noir ou blanc). Le fond en gris pâle représente les données non simulées. Au départ, l'image contient quatre données conditionnantes représentées par des encadrés en traits gras. La simulation a déjà simulé 16 pixels et leur a attribué la valeur 0 ou 1. Le pixel gris au centre est le pixel présentement simulé. La valeur que nous attribuons à ce

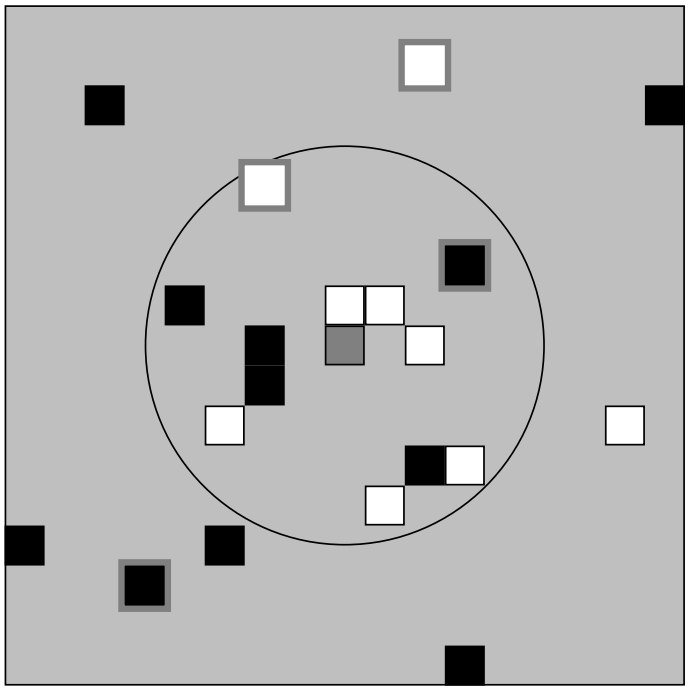


Figure 1.1 Simulation du pixel central en gris foncé conditionnellement aux douze plus proches voisins déjà simulés.

pixel est déterminée en fonction du voisinage, c'est-à-dire en fonction des 12 pixels contenus dans le cercle.

L'attribution de la valeur de chaque pixel simulé est une étape critique. En effet, nous voulons que les choix des valeurs des pixels permettent de reproduire le plus fidèlement possible les propriétés statistiques d'un milieu donné (un réseau de chenaux, par exemple). Nous devons donc évaluer quelle serait la valeur la plus probable, étant donné les valeurs prises par les voisins.

Le nombre de voisins conditionnant le choix de la valeur du pixel simulé peut se limiter aux pixels inclus dans un rayon de portée maximale tel qu'illustré par le cercle de la figure 1.1. Ce rayon est habituellement déterminé par la distance de corrélation qui correspond à la distance minimale h_c à partir de laquelle les valeurs $Z(n)$ et $Z(m)$ de deux pixels sont non corrélées. Ainsi, pour deux pixels espacés d'une distance d'au moins h_c, nous avons que $\mathrm{Cov}\left[Z(n), Z(m)\right] = 0$. Pour limiter le temps de calcul, nous pouvons également limiter le nombre de pixels voisins à un nombre maximal fixe.

1.2 Conditionnement à l'aide du krigeage

Une méthode d'interpolation qui permet d'estimer la valeur d'un pixel donné en fonction des pixels voisins est le krigeage. Dans le krigeage, la valeur estimée est une combinaison linéaire des valeurs des pixels voisins. Deux types de krigeage importants sont le krigeage simple où la moyenne des données est connue et le krigeage ordinaire où la moyenne est inconnue. Considérons le krigeage ordinaire. Dans ce dernier, la valeur estimée $\hat{Z}$ d'un pixel est une combinaison linéaire des valeurs U_ℓ de m pixels voisins, c'est-à-dire que

$$\hat{Z} = \sum_{\ell=1}^{m} \lambda_\ell \, U_\ell, \tag{1.1}$$

où les λ_ℓ respectent

$$\sum_{\ell=1}^{m} \lambda_\ell = 1, \tag{1.2}$$

pour que $\mathrm{E}\left[\hat{Z}\right] = \mathrm{E}\left[U_\ell\right] = \mathrm{E}\left[U_1\right], \forall \ell$, dans le cas où le champ aléatoire est homogène statistiquement. Dans la figure 1.1, par exemple, la valeur estimée du pixel central se fait en fonction des $m = 12$ plus proches voisins. Les coefficients λ_ℓ sont choisis afin de minimiser la variance d'estimation

$$\sigma_{\mathrm{e}}^2 = \mathrm{Var}\left[Z - \hat{Z}\right], \tag{1.3}$$

où Z est la vraie valeur (inconnue) du pixel. Pour trouver les λ_ℓ optimaux, la méthode du krigeage demande de connaître à priori les covariances $\mathrm{Cov}\left[Z, U_j\right]$. Pour un champ aléatoire donné, nous adoptons généralement l'hypothèse de stationnarité qui suppose une moyenne constante du champ aléatoire et qui suppose que les covariances entre deux points ne dépendent que du vecteur de distance $\boldsymbol{h}$ séparant les deux points. Si en plus le champ est isotrope, les covariances $\mathrm{Cov}\left[Z, U_\ell\right]$ ne dépendent alors que de la distance euclidienne h entre les deux points.

En utilisant les λ_ℓ qui minimisent la variance d'estimation donnée en (1.3), nous obtenons (Chilès et Delfiner, 1999), la variance du krigeage ordinaire

$$\sigma_{\mathrm{ko}}^2 = \mathrm{Var}\left[Z\right] - \sum_{\ell=1}^{m} \lambda_\ell \, \mathrm{Cov}\left[Z, U_\ell\right] - \mu, \tag{1.4}$$

où μ est le multiplicateur de Lagrange utilisé pour trouver les λ_ℓ optimaux. Ainsi, la méthode du krigeage fournit un estimé $\hat{Z}$ de Z et nous renseigne sur l'erreur associée à cet estimé. De plus, la variance σ^2_{ko} ne dépend que de la configuration spatiale des m voisins et non de leurs valeurs.

L'estimation par krigeage est utilisée dans les simulations séquentielles gaussiennes telles que *sgsim* (Deutsch et Journel, 1998). La méthode *sgsim* est une méthode de simulation de type point par point car les pixels sont simulés un à la fois et de façon séquentielle. Dans cette simulation, nous commençons par fixer un chemin aléatoire. Ce chemin est défini par une permutation aléatoire des pixels à simuler. Chaque pixel n'est visité qu'une seule fois le long du chemin et se voit attribuer une valeur. Cette valeur est choisie au hasard conditionnellement aux pixels existants. Plus précisément, la valeur S du pixel simulé est pigée au hasard selon une loi normale de moyenne égale à l'estimateur du krigeage et de variance égale à la variance d'estimation du krigeage. Ainsi, dans le cas du krigeage ordinaire, nous avons

$$S \sim \text{N}(\hat{Z}, \sigma^2_{ko}). \tag{1.5}$$

En utilisant le krigeage comme estimateur, les simulations séquentielles gaussiennes permettent de reproduire les statistiques deux-points. Ainsi, pour un champ gaussien, la covariance entre deux pixels simulés $\text{Cov}\left[S(n), S(m)\right]$ correspondra à la covariance attendue et, dans le cas du krigeage simple, l'espérance d'un pixel simulé correspondra à la moyenne attendue.

Nous utilisons aussi le *variogramme* pour modéliser la covariance entre deux points dans un champ aléatoire. Soit un champ aléatoire où $Z(n)$ et $Z(m)$ sont les valeurs de deux pixels séparés par le vecteur $\boldsymbol{h}$, le variogramme $\gamma(\boldsymbol{h})$ correspond à $\gamma(\boldsymbol{h}) = \text{Var}\left[Z(n) - Z(m)\right]/2$. Si le champ est isotrope, le variogramme est uniquement fonction de la distance h entre deux pixels. Dans ce cas, on démontre facilement pour un champ stationnaire que $\gamma(h) = \text{Var}\left[Z\right] - \text{Cov}\left[Z(n), Z(m)\right]$. Le variogramme est plus général que la covariance puisqu'il peut exister même si σ^2 n'est pas défini (dans un champ non stationnaire, par exemple). De plus, son estimation est plus simple puisqu'il ne demande pas, contrairement à la covariance, de connaître ou d'estimer la moyenne du champ (Chilès et Delfiner, 1999, p. 30).

1.3 Les simulations multi-points, point par point

Bien que les simulations gaussiennes reproduisent efficacement les statistiques deux-points, elles ne permettent pas de reproduire les informations plus structurelles d'un champ aléatoire (Journel, 1993 ; Strebelle, 2000 ; Saucier, Richer, Muller, 2002). Le réseau de chenaux de la figure 1.2, par exemple, présente des courbes, des interconnections et une structure qui ne peuvent pas être modélisées uniquement avec la covariance entre deux points.

Dans ce qui suit, les m valeurs U_ℓ des pixels voisins sont réunies dans le vecteur aléatoire $\boldsymbol{U}$ que nous appelons l'*état* $\boldsymbol{U}$. Au lieu d'estimer par une combinaison linéaire des U_ℓ la valeur S à attribuer au pixel simulé, nous cherchons quelle(s) valeur(s) le pixel simulé peut prendre étant donné que l'état $\boldsymbol{U}$ est observé. Pour faire ce choix, nous devons connaître, pour notre champ aléatoire, la probabilité que S prenne la valeur s_k étant donné que l'état $\boldsymbol{U}$ se trouve dans l'état $\boldsymbol{u}_j$. Nous appelons *probabilités de transition* les probabilités conditionnelles

$$p_{j,k} := \mathrm{P}\left[S = s_k \;\middle|\; \boldsymbol{U} = \boldsymbol{u}_j\right]. \tag{1.6}$$

Comme $\boldsymbol{U}$ est constitué d'un nombre $m \geq 2$ de valeurs de pixels, le choix de S tiendra compte des statistiques dites *multi-points*. Pour que les probabilités (1.6) ne soient pas nulles, les valeurs possibles de s_k et de $\boldsymbol{u}_j$ doivent être limitées à un ensemble fini. Dans cette thèse, nous supposons que les pixels ne peuvent prendre que les valeurs 0 (noir) ou 1 (blanc), nous avons donc $s_k \in \{0, 1\}$ et $\boldsymbol{u}_j \in \{0, 1\}^m$.

Figure 1.2 Réseau de chenaux.

Afin d'estimer les probabilités (1.6), Guardiano et Srivastava (1993) ont proposé d'utiliser une *image de référence* comme source d'information. L'image de référence est supposée être un échantillon représentatif du champ aléatoire étudié.

Pour estimer les probabilités de transition à l'aide d'une image de référence, nous commençons par réécrire (1.6) sous la forme

$$p_{j,k} = \frac{\mathrm{P}\left[S = s_k, \boldsymbol{U} = \boldsymbol{u}_j\right]}{\mathrm{P}\left[\boldsymbol{U} = \boldsymbol{u}_j\right]}. \tag{1.7}$$

Le rapport du terme de droite de l'équation (1.7) est estimé avec

$$\frac{\mathrm{P}\left[S = s_k, \boldsymbol{U} = \boldsymbol{u}_j\right]}{\mathrm{P}\left[\boldsymbol{U} = \boldsymbol{u}_j\right]} \simeq \frac{c_k(\boldsymbol{u}_j)}{c(\boldsymbol{u}_j)}, \tag{1.8}$$

où $c(\boldsymbol{u}_j)$ et $c_k(\boldsymbol{u}_j)$ sont les nombres d'occurrences dans l'image de référence de l'évènement $\boldsymbol{U} = \boldsymbol{u}_j$ et du nombre d'occurrences de l'évènement $\{S = s_k\} \cap \{\boldsymbol{U} = \boldsymbol{u}_j\}$ respectivement. Le comptage de ces évènements se fait en scannant toute l'image de référence. Par exemple, dans la petite image de référence présentée à la figure 1.3, les compteurs associés à l'état $\boldsymbol{u}_j = (1, 1, 0, 0, 1)$ sont $c(\boldsymbol{u}_j) = 2$, $c_0(\boldsymbol{u}_j) = 1$ et $c_1(\boldsymbol{u}_j) = 1$ et les probabilités de transition sont $p_{j,0} = 1/2$ et $p_{j,1} = 1/2$. Puisque s_k ne peut valoir que 0 ou 1, les compteurs d'une configuration $\boldsymbol{u}_j$ de voisins doivent respecter $c_0(\boldsymbol{u}_j) + c_1(\boldsymbol{u}_j) = c(\boldsymbol{u}_j)$, c'est-à-dire que toute copie du voisinage $\boldsymbol{u}_j$ observée dans l'image de référence doit nécessairement être associée à un pixel central s_k de valeur 0 ou 1. Pour que l'approximation (1.8) soit valide, il faut que l'image de référence soit représentative du champ aléatoire et suffisamment grande pour que les états $\boldsymbol{u}_j$ puissent être observés un grand nombre de fois, c'est-à-dire que nous voulons que les compteurs satisfassent $c(\boldsymbol{u}_j) \gg 1$. Par exemple, Guardiano et Srivastava utilisent le seuil $c(\boldsymbol{u}_j) \geq 20$. Ainsi, si le compteur $c(\boldsymbol{u}_j)$ est inférieur à 20, Guardiano et Srivastava proposent de diminuer le nombre de voisins m considérés jusqu'à ce que $c(\boldsymbol{u}_j) \geq 20$.

La simulation multi-point décrite par Guardiano et Srivastava, appelée *enesim* pour *extended normal equation simulation*, est une simulation point par point similaire aux méthodes de simulation gaussienne. Tout comme dans les simulations gaussiennes, nous fixons un chemin aléatoire et chaque pixel est visité une fois en suivant ce chemin. La particularité de la méthode *enesim* est la façon de choisir la valeur attribuée à chaque pixel. Cette valeur est

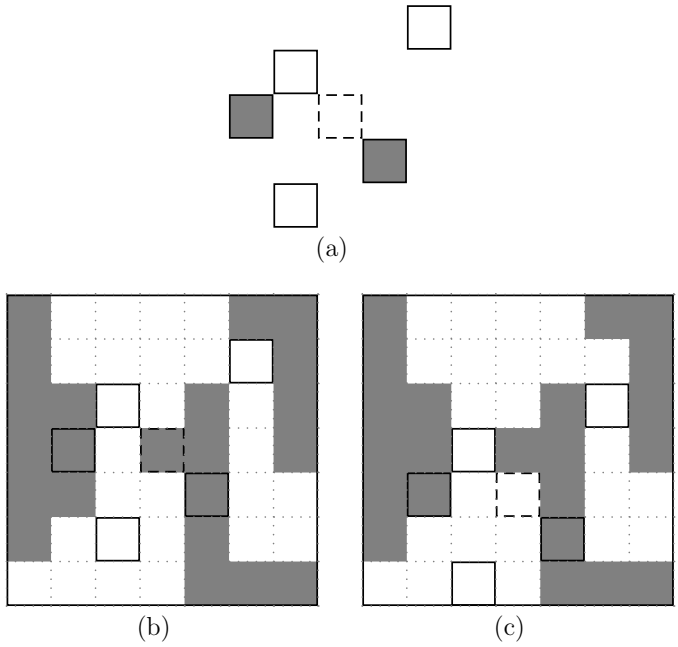


Figure 1.3 (a) Un voisinage de cinq pixels $\boldsymbol{U} = \boldsymbol{u}_j = (1, 1, 0, 0, 1)$ avec le pixel central S en tireté à simuler. (b) Le seul pixel (tireté) de valeur $s_0 = 0$ observé avec le voisinage $\boldsymbol{u}_j$. (c) Le seul pixel (tireté) de valeur $s_1 = 1$ observé avec le voisinage $\boldsymbol{u}_j$.

choisie au hasard en utilisant les probabilités de transition $p_{j,k}$ de l'équation (1.6). Cette modification fait en sorte que l'image construite a tendance à produire des structures semblables à celles observées dans l'image de référence.

1.4 La méthode de simulation *snesim*

L'inconvénient de la méthode de simulation *enesim* de Guardiano et Srivastava est le temps de calcul requis pour estimer les probabilités de transition $p_{j,k}$ de l'équation (1.6). En effet, pour estimer les $p_{j,k}$, il faut compter les $c(\boldsymbol{u}_j)$ et $c_k(\boldsymbol{u}_j)$ à chaque étape de la simulation. Si l'on considère des images de référence de taille d'au moins 100×100 pixels, cette recherche de tous les $\boldsymbol{u}_j$ devient coûteuse en temps de calcul et la méthode devient impraticable.

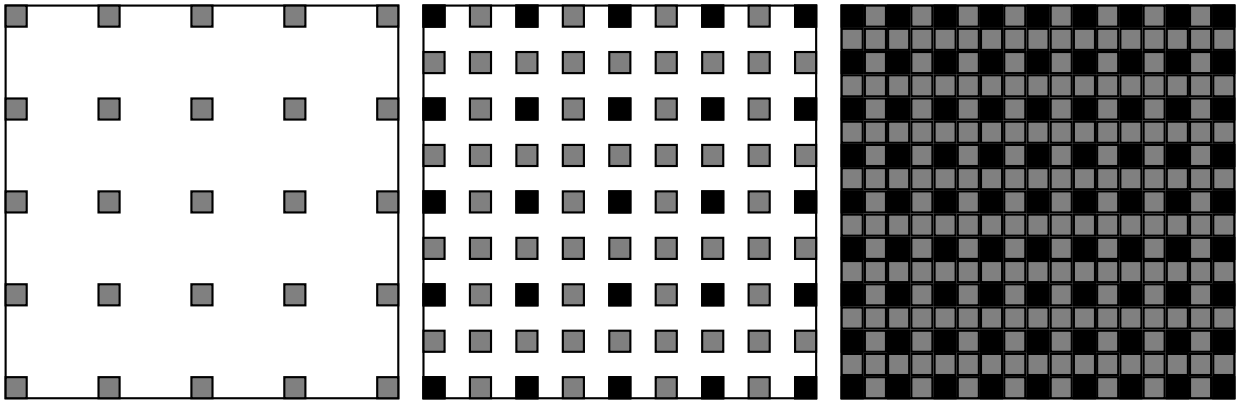
Strebelle (2000) a proposé une version améliorée de la méthode de Guardiano et Srivastava appelée *snesim* (*single normal equation simulation*). La principale amélioration apportée par la méthode *snesim* est la manière d'évaluer les compteurs $c(\boldsymbol{u}_j)$ et $c_k(\boldsymbol{u}_j)$ permettant d'estimer les probabilités $p_{j,k}$. Dans sa méthode, Strebelle fixe d'abord un *gabarit* (*data template*) pour

les états $\boldsymbol{U}$. Le gabarit est l'ensemble des positions relatives des voisins U_ℓ formant l'état $\boldsymbol{U}$. Ensuite, l'image de référence est scannée une fois et tous les états possibles $\boldsymbol{u}_j$ de l'image de référence sont enregistrés dans un arbre de recherche suivant le gabarit des états $\boldsymbol{U}$. Les états $\boldsymbol{u}_j$ y sont enregistrés avec leurs compteurs $c(\boldsymbol{u}_j)$ et $c_k(\boldsymbol{u}_j)$. Bien que l'initialisation de l'arbre de recherche prenne un certain temps, l'usage de l'arbre rend les simulations moins coûteuses en temps de calcul, car il évite de parcourir à nouveau l'image de référence.

Une autre amélioration algorithmique apportée par la méthode *snesim* est l'usage d'une structure multi-échelle pour définir le chemin de la simulation point par point. L'ensemble des pixels est divisé en plusieurs *grilles* qui forment une partition de l'ensemble des pixels de l'image simulée (voir figure 1.4). Cette structure est appelée *multiple grid*. Chaque grille possède son gabarit (*data template*) pour le voisinage. La simulation commence par la grille où les pixels sont le plus espacés (figure 1.4a) et elle enchaîne ensuite par la simulation des grilles intermédiaires (figure 1.4b). Finalement, la simulation termine avec la grille la plus fine qui contient tous les pixels sauf ceux des échelles déjà simulées (figure 1.4c).

1.5 La méthode de simulation par échantillonnage direct

Plus récemment, Mariethoz, Renard et Straubhaar (2010) ont proposé une méthode semblable à la simulation point par point *enesim* de Guardiano et Srivastava. Cette dernière, appelée *direct sampling method* (DS), a la particularité de ne pas nécessiter le calcul des compteurs $c(\boldsymbol{u}_j)$ et $c_k(\boldsymbol{u}_j)$ afin d'évaluer les probabilités de transition $p_{j,k}$. En effet, étant donné un voisinage de pixels $\boldsymbol{u}_j$ tel que celui de la figure 1.3a, nous choisissons d'abord un emplacement au hasard et de manière équiprobable dans l'image de référence. Ensuite, l'image de référence est balayée de manière unilatérale jusqu'à ce que l'état $\boldsymbol{u}_j$ soit observé. On attribue alors la valeur centrale observée s_k. L'avantage de la méthode DS est qu'elle évite d'avoir à emmagasiner toute l'information des états $\boldsymbol{u}_j$ dans un arbre de recherche. En effet, dans la méthode *snesim* de Strebelle, la taille du gabarit utilisé, le nombre de faciès du champ simulé et l'entropie de l'image de référence influencent directement sur la taille de l'arbre de recherche et l'espace mémoire utilisé par l'algorithme.



(a) Grande échelle (b) Échelle intermédiaire (c) Échelle fine

Figure 1.4 Exemple de multi-grille de l'algorithme *snesim*. (a) Dans la première grille, les pixels à simuler sont en gris. (b) Dans la grille intermédiaire, les pixels en noir ont été fixés par la première échelle et les pixels en gris sont à simuler. (c) Dans la dernière grille, les pixels en noir ont été simulés et les pixels en gris restants sont à simuler.

1.6 Les simulations par rapiéçage de motifs

Certaines méthodes de simulation d'image sont davantage axées sur la reconnaissance de formes. Ces méthodes visitent chaque noeud d'une grille prédéterminée et viennent y coller le *motif* qui s'y agence le mieux (voir figure 1.5). Un motif consiste typiquement en un morceau carré d'image de taille $L \times L$ pixels prélevés dans une image de référence selon certains critères d'agencement ou de contrôle des statistiques. Ainsi, dans une simulation par rapiéçage, l'important n'est plus nécessairement les probabilités de transition, mais plutôt les critères d'agencement pour que l'image résultante de ce casse-tête soit continue et n'ait pas l'air d'un collage de morceaux choisis au hasard dans l'image de référence. Dans ce qui suit, les $L \times L$ valeurs des pixels du motif collé à l'image sont réunies dans le vecteur aléatoire $\boldsymbol{S}$ que nous appelons l'*état* $\boldsymbol{S}$. Cet état est choisi conditionnellement à un ensemble de données voisines réunies dans le vecteur aléatoire $\boldsymbol{U}$ appelé l'état $\boldsymbol{U}$.

Un premier exemple de simulation par rapiéçage de motifs est la *simulation conditionnelle avec motifs*, ou en anglais *Conditional Simulation with Patterns* (*simpat*), de Arpat et Caers (2005; 2007). La méthode *simpat* utilise une structure multi-échelle semblable à la multi-grille de la méthode *snesim* (figure 1.4), c'est-à-dire que nous commençons par simuler la grande

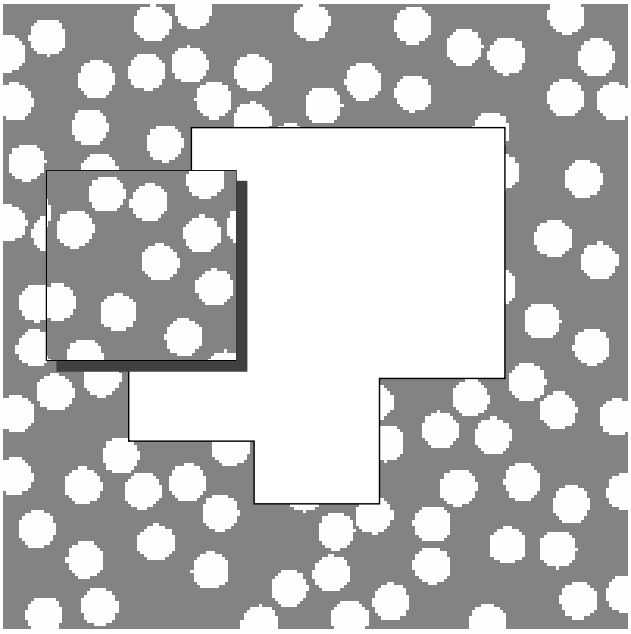


Figure 1.5 Ajout de données (l'état $\boldsymbol{S}$) dans une simulation par rapiéçage de motifs.

échelle, puis nous enchaînons avec les échelles inférieures.

Considérons la simulation de la grande échelle de la méthode *simpat*. Nous commençons par choisir un chemin aléatoire visitant tous les nœuds de la grille à grande échelle (grille semblable à celle de la figure 1.4a). Lors de la visite de chaque nœud, au lieu de ne simuler que le pixel central à l'aide des probabilités $p_{j,k}$, nous allons coller un *motif* de taille $L \times L$ (l'état $\boldsymbol{S}$) pigé dans l'image de référence conditionnellement aux pixels déjà simulés dans la grande échelle (conditionnellement à un état $\boldsymbol{U}$). Un exemple de motif utilisé à grande échelle est illustré à la figure 1.6a. Dans cette figure, le motif est de taille 9×9. L'état $\boldsymbol{S}$ contient donc les valeurs de 9×9 pixels. Seuls les pixels gris correspondant à des pixels de la grande grille sont utilisés dans la recherche du motif de l'image de référence à coller. Ainsi, l'état $\boldsymbol{U}$ ne contient que les valeurs des neuf pixels en gris. Cependant, toutes les 9×9 valeurs de l'état $\boldsymbol{S}$ sont collées à l'image simulée.

À la suite de la simulation de la grande échelle, la simulation continue avec la ou les échelle(s) intermédiaire(s). En reprenant notre exemple, nous parcourons la grille intermédiaire de la figure 1.4b en utilisant un motif 5×5 tel qu'à la figure 1.6b. Cette simulation affine les détails du champ aléatoire. Nous procédons de même avec la grille fine, c'est-à-dire

que nous parcourons la grille de la figure 1.4c avec le motif 3×3 de la figure 1.6c.

Comme dans toute technique de simulation séquentielle, l'étape critique est le choix des données ajoutées (l'état $\boldsymbol{S}$) conditionnellement aux données existantes (l'état $\boldsymbol{U}$). Si $\mathcal{S}_r$ est l'ensemble de tous les états $\boldsymbol{s}$ de taille $L \times L$ observables dans l'image de référence, l'état $\boldsymbol{S}$ est alors pris parmi un des états $\boldsymbol{s} \in \mathcal{S}_r$ ayant le plus de pixels communs avec $\boldsymbol{U}$. Plus précisément, soit $\boldsymbol{u}(\boldsymbol{s})$ le sous-ensemble des éléments d'un état $\boldsymbol{s}$ associés aux positions des pixels d'un état $\boldsymbol{u}$, l'état $\boldsymbol{S}$ est choisi dans l'ensemble

$$A(\boldsymbol{U}) := \underset{\boldsymbol{s} \in \mathcal{S}_r}{\operatorname{argmin}} \quad d(\boldsymbol{U}, \boldsymbol{u}(\boldsymbol{s})), \tag{1.9}$$

où nous utilisons la distance de Manhattan

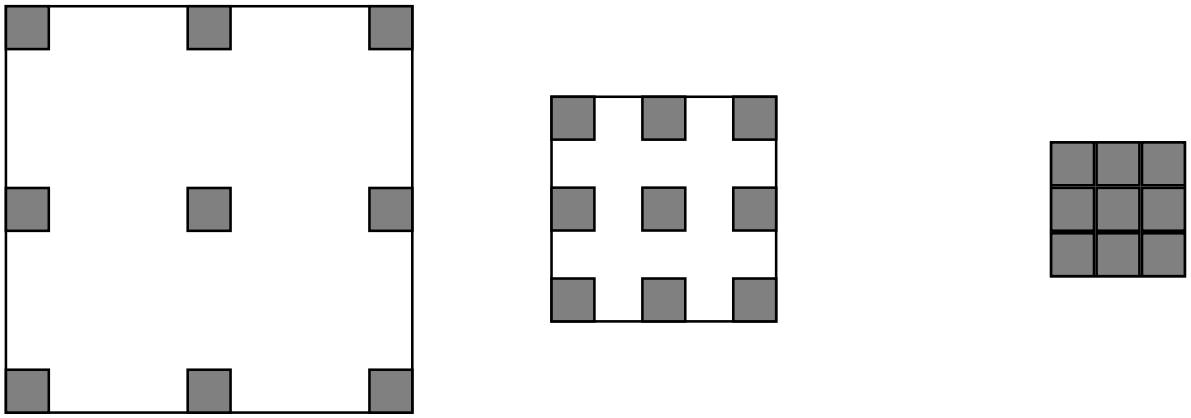
$$d(\boldsymbol{u}_1, \boldsymbol{u}_2) = \sum_{\ell \in \mathcal{L}} |u_1(\ell) - u_2(\ell)| \,. \tag{1.10}$$

Dans l'équation (1.10), $u_1(\ell)$ est le $\ell^{\text{ième}}$ élément du vecteur $\boldsymbol{u}_1$ et la somme se fait sur les indices $\ell \in \mathcal{L}$, où $\mathcal{L}$ est l'ensemble des indices des pixels déjà simulés qui sont dans le support de $\boldsymbol{u}_1$. Lorsqu'il y a présence de données conditionnantes, la méthode de Arpat et Caers limite le choix de $\boldsymbol{S}$ à un état $\boldsymbol{s} \in \mathcal{S}_r$ qui respecte les données conditionnantes incluses dans le cadre de $\boldsymbol{U}$. Si nous ne pouvons pas trouver d'état $\boldsymbol{s}$ qui respecte toutes les données conditionnantes dans le cadre de $\boldsymbol{U}$, l'état qui en respecte le plus est choisi. Cette situation se produit si l'image de référence n'est pas représentative du champ aléatoire étudié.

Nous pouvons faire le lien avec les probabilités de transition des méthodes précédentes point par point en considérant les probabilités

$$\begin{aligned} p_{j,k} &= \mathrm{P}\left[\boldsymbol{S} = \boldsymbol{s}_k \,\middle|\, \boldsymbol{U} = \boldsymbol{u}_j\right] \\ &= \begin{cases} \dfrac{1}{|A(\boldsymbol{u}_j)|}, & \text{si } \boldsymbol{s}_k \in A(\boldsymbol{u}_j), \\ 0, & \text{sinon.} \end{cases} \end{aligned} \tag{1.11}$$

Ces dernières probabilités permettent de définir le choix du morceau de l'image de référence $\boldsymbol{s}_k$ en fonction des données simulées $\boldsymbol{u}_j$. Par contre, dans certaines applications, l'ensemble $A(\boldsymbol{u}_j)$



(a) Grand motif (b) Motif intermédiaire (c) Motif fin

Figure 1.6 Exemple de motifs dans la simulation *simpat* de Arpat et Caers. (a) Motif utilisé pour la grande échelle. Les pixels espacés en gris sont ceux utilisés pour trouver le motif le plus proche (état $\boldsymbol{u}$), mais tous les pixels à l'intérieur du cadre sont collés à l'image simulée (état $\boldsymbol{s}$). (b) Motif intermédiaire. (c) Dernier motif utilisé pour affiner l'image simulée.

pourrait se limiter à un seul état $\boldsymbol{s}$. Dans ce cas, le choix devient déterministe et l'analyse stochastique n'a pas lieu d'être. Ceci arrive typiquement quand l'image de référence présente une grande variété d'états et qu'un état $\boldsymbol{u}$ donné ne présente pas ou peu d'occurrences dans l'image de référence. La simulation prend alors l'état $\boldsymbol{s}_k$ de l'image de référence qui s'agence le mieux avec la distance de Manhattan.

Dans la méthode *simpat*, l'utilisation de la distance de Manhattan ne permet pas nécessairement aux morceaux collés de bien s'agencer les uns avec les autres. Pour minimiser l'introduction de discontinuités lors du collage, Arpat et Caers ont suggéré d'appliquer une *transformée de distances* à l'image de référence. Les motifs ainsi transformés sont alors plus simples à agencer.

1.7 La simulation unilatérale PSM

Les méthodes de simulation dites *unilatérales* sont celles qui, au lieu de suivre un chemin aléatoire, procèdent en balayant l'image simulée dans un ordre linéaire. Ainsi, la méthode de simulation de Parra et Ortiz (2011) est une simulation point par point qui débute dans le

coin inférieur gauche et balaye tous les pixels de l'image simulée de gauche à droite et de bas en haut.

Une autre méthode unilatérale est la *méthode de simulation par rapiéçage de motif*, ou en anglais *Patchwork Simulation Method* (*PSM*), de El Ouassini, Saucier, Marcotte et Favis (2006; 2008). Cette dernière, tout comme la méthode *simpat* de Arpat et Caers, procède en collant des morceaux carrés d'une image de référence de taille $L \times L$. Par contre, la méthode de El Ouassini *et al.* procède de manière unilatérale, c'est-à-dire que le chemin de la simulation avance en balayant l'image simulée de gauche à droite et de haut en bas plutôt que d'être aléatoire comme la méthode *simpat*. Comme il est montré à la figure 1.7, à chaque étape de la simulation un morceau est collé en fonction des données voisines. Pour déterminer le choix du morceau à coller, El Ouassini *et al.* utilisent les probabilités de transition

$$p_{j,k} = \mathrm{P}\left[\boldsymbol{S} = \boldsymbol{s}_k \,\middle|\, \boldsymbol{U} = \boldsymbol{u}_j\right], \tag{1.12}$$

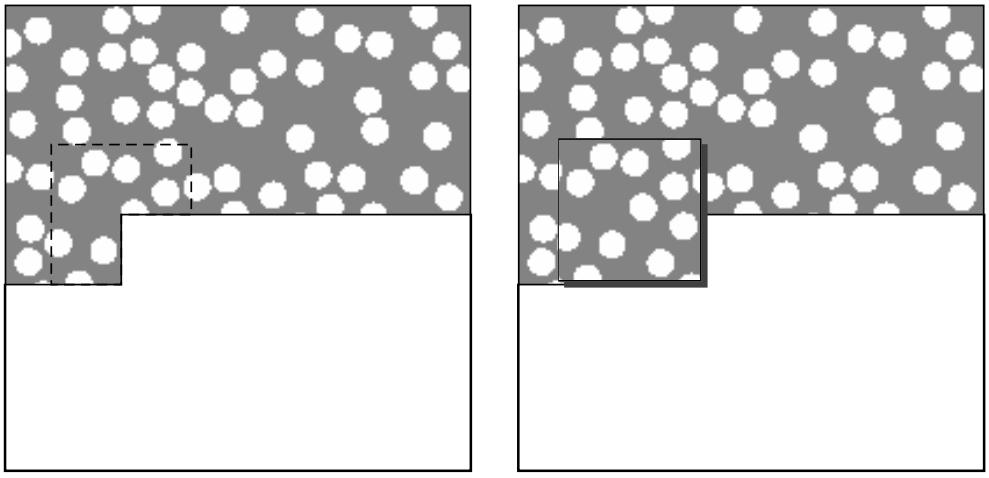
où $\boldsymbol{U}$ est appelé l'état de départ (le voisinage utilisé) et $\boldsymbol{S}$ est appelé l'état d'arrivée (les données simulées). Pour évaluer les probabilités de transition, l'image de référence est scannée pour trouver les occurrences de $\boldsymbol{u}_j$ associées à $\boldsymbol{s}_k$. Un état quelconque en forme de L (figure 1.7a) est dit de type $\boldsymbol{u}$ et un état carré quelconque (figure 1.7b) est dit de type $\boldsymbol{s}$.

Par son formalisme, la simulation unilatérale PSM ressemble à un champ aléatoire de Markov (*Markov mesh random field*) (Pickard, 1980; Razlighi *et al.*, 2009). Un champ aléatoire de Markov est l'extension 2D d'une chaîne de Markov. La chaîne de Markov est un processus aléatoire (1D) à temps discret dans lequel une variable aléatoire $X(n)$ mesurée à l'instant n ne dépend que de l'instant précédent $X(n-1)$. Plus précisément, nous avons

$$\mathrm{P}\left[X(n) = x_n \;\middle|\; \bigcap_{j=0}^{n-1} \{X(j) = x_j\}\right] = \mathrm{P}\left[X(n) = x_n \,\middle|\, X(n-1) = x_{n-1}\right]. \tag{1.13}$$

De manière similaire, dans la simulation unilatérale PSM, l'état $\boldsymbol{S}$ ne dépend que des données précédemment simulées ($\boldsymbol{U}$ est constitué des données à gauche et en haut).

Lorsque l'état $\boldsymbol{U} = \boldsymbol{u}_j$ n'est pas observé dans l'image de référence, la méthode PSM utilise une métrique particulière pour trouver un des états $\boldsymbol{u}$ de l'image de référence qui est le plus



(a) État de départ $\boldsymbol{U}$ (b) État d'arrivée $\boldsymbol{S}$

Figure 1.7 Ajout de données dans la simulation unilatérale PSM. (a) L'état de départ $\boldsymbol{U}$ est le voisinage utilisé pour simuler le carré à ajouter. (b) L'état d'arrivée $\boldsymbol{S}$ représente les données ajoutées.

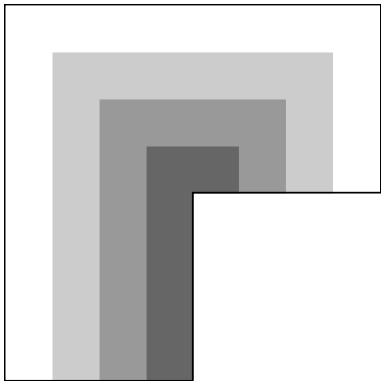


Figure 1.8 Distribution spatiale des poids $w(\ell)$ pour la distance $d(\boldsymbol{u}_1, \boldsymbol{u}_2)$ dans la simulation unilatérale PSM. Ici, le gabarit de l'état $\boldsymbol{u}$ est de taille 8×8 pixels et les poids varient linéairement de 1 (en gris foncé) à 4 (en blanc).

proche de l'état $\boldsymbol{u}_j$. Étant donné $\mathcal{U}_r$ l'ensemble de tous les états de type $\boldsymbol{u}$ observables dans l'image de référence, nous choisissons un état $\boldsymbol{u}$ dans

$$A(\boldsymbol{u}_j) = \underset{\boldsymbol{u}\in\mathcal{U}_r}{\operatorname{argmin}}\, d(\boldsymbol{u}, \boldsymbol{u}_j), \tag{1.14}$$

où la distance d est définie par

$$d(\boldsymbol{u}_1, \boldsymbol{u}_2) = \sum_{\ell=1}^{m} w_\ell \left| u_1(\ell) - u_2(\ell) \right|. \tag{1.15}$$

La particularité de cette distance est l'introduction des poids w_ℓ. Ces poids sont attribués en fonction de la position spatiale des pixels ℓ telle qu'illustrée à la figure 1.8. Cette distribution des poids permet de mettre l'accent au pourtour des données ajoutées afin d'assurer la continuité avec les données existantes.

Récemment, Tahmasebi *et al.* (2012) ont proposé une approche semblable à la simulation unilatérale PSM. La particularité de leur méthode est d'utiliser comme métrique la corrélation croisée pour trouver l'état $\boldsymbol{u}$ qui s'agence le mieux dans l'image de référence.

1.8 Le contrôle des statistiques

Lors d'une simulation d'image, nous voulons pouvoir garder le contrôle des statistiques un-points et deux-points telles que la moyenne, la variance ou encore la distribution de la moyenne locale. En effet, si nous laissons rouler une simulation qui ne fait qu'agencer des morceaux de casse-tête, la méthode de simulation pourrait juger que les morceaux les plus faciles à agencer sont les morceaux tout noirs et ainsi ne produire que des images d'un noir uniforme. Considérons, par exemple, la simulation *simpat*. Puisque la méthode ne repose pas explicitement sur une méthodologie impliquant un processus stochastique et un contrôle direct des statistiques, rien n'assure que les statistiques un-points telles que la valeur moyenne des pixels seront respectées. Bien que le collage des morceaux à grande échelle assure que l'image simulée représentera un aspect global semblable à l'image de référence, rien ne garantit que les proportions de pixels blancs et noirs seront respectées. En effet, s'il y a un biais dès la première échelle, la simulation le traînera tout au long de la simulation.

Dans l'algorithme *snesim* de Strebelle, il existe une méthode de contrôle appelée *servo system* qui ajuste les probabilités $p_{j,k}$ afin de conserver le ratio désiré de pixels blancs et noirs. Dans une autre version de l'algorithme *snesim* présentée par Boucher et Stright (2008), l'ensemble des états de l'échelle fine est séparé en une partition. L'appartenance d'un état à une des classes de la partition se fait en fonction de mesures faites à une plus grande échelle. De cette façon, la simulation à l'échelle fine est contrainte de respecter les mesures faites à plus grande échelle (la connectivité ou la moyenne locale, par exemple).

Une autre approche multi-point présentée par Chugunova (2008) permet de réaliser des simulations non stationnaires, c'est-à-dire que les propriétés statistiques ne sont pas constantes mais varient en fonction de leur position dans l'image. Dans sa méthode, appelée *simulation multi-point non stationnaire*, la valeur attribuée à chaque pixel est conditionnelle à une *donnée auxiliaire*. Les données auxiliaires sont définies pour tous les emplacements de l'image simulée et fournissent de l'information sur la non-stationnarité de l'image. Nous pourrions par exemple choisir de mettre dans les données auxiliaires l'espérance attendue d'un pixel en fonction de sa position et ainsi simuler une image présentant un gradient : les pixels à gauche de l'image étant plus blancs et ceux à droite de l'image étant plus noirs. Nous pourrions aussi choisir d'y mettre l'orientation attendue de chenaux simulés et ainsi changer leur orientation. Arpat et Caers (2005) présentent également une méthode de simulation où les motifs sont collés conditionnellement à une grille de données auxiliaires appelées *soft data*. Ces soft data sont en fait de l'information à priori sur le milieu étudié. Elles sont obtenues en appliquant un filtre passe-bas au milieu étudié (convolution avec un boîte $L \times L$). Pour tenir compte de ces soft data, l'image de référence est dédoublée avec, d'une part, l'image de référence originale et, d'autre part, l'image de référence filtrée par convolution avec un carré $L \times L$. Lors de la synthèse, le rapiéçage se fait en calculant la distance par rapport aux données simulées et par rapport aux soft data.

CHAPITRE 2

DÉMARCHE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE

2.1 La métrique

Les deux méthodes de simulation développées dans cette thèse font suite à la méthode de simulation *patchwork simulation method* (PSM) de El Ouassini, Saucier, Marcotte et Favis (2006; 2008) décrite à la section 1.7. Un aspect important de cette méthode PSM est l'utilisation d'une métrique non uniforme donnant plus de poids aux pixels se trouvant proches du pourtour de l'état collé $\boldsymbol{S}$ (voir figures 1.7 et 1.8). Ainsi, lorsque l'état de départ $\boldsymbol{U}$ (figure 1.7a) n'est pas observé dans l'image de référence cette métrique permet de trouver un état qui s'agence bien avec les pixels à proximité de la bordure. De cette façon, l'image simulée présente une meilleure continuité et par le fait même un meilleur aspect visuel. Dans le chapitre 3, nous présentons une métrique améliorée qui non seulement donne de l'importance aux pixels à proximité du pourtour du morceau collé mais en donne aussi aux pixels qui se trouvent au-delà du pourtour. Cette métrique est utilisée par nos deux méthodes de simulation par rapiéçage de motifs.

2.2 La méthode unilatérale PSM

À l'aide de cette métrique améliorée, nous avons développé une nouvelle méthode de rapiéçage unilatérale appelée *patchwork simulation method* (PSM) comme la version originale de El Ouassini *et al.* Cette version améliorée, décrite en détail au chapitre 4, est unilatérale, c'est-à-dire qu'elle procède de gauche à droite et de haut en bas en collant dans l'image simulée des morceaux prélevés dans l'image de référence. L'image simulée n'est parcourue qu'une seule fois et le collage des morceaux se fait en utilisant les probabilités de transition $p_{j,k} = \mathrm{P}\left[\boldsymbol{S}(n) = \boldsymbol{s}_k \,\middle|\, \boldsymbol{U}(n) = \boldsymbol{u}_j\right]$ décrites à la section 1.7.

En plus de la nouvelle métrique, nous avons développé une méthode de contrôle de la moyenne locale. Ce contrôle s'effectue de façon semblable au partitionnement de Boucher et

Stright (2008), c'est-à-dire que les états de type $\boldsymbol{s}$ observés dans l'image de référence sont classés dans une partition. L'appartenance à une des classes de la partition est déterminée par la valeur moyenne des pixels observés dans le quartier inférieur droit de l'état $\boldsymbol{s}$, c'est-à-dire dans la région qui ajoute des données à la simulation. Grâce à cette partition, nous sommes en mesure de choisir le morceau à coller afin de contrôler la distribution de la moyenne locale ou, plus précisément, afin de contrôler l'*histogramme de la moyenne locale*. L'utilisation de cette partition permet un contrôle qui va au-delà de la simple reproduction de l'histogramme de l'image de référence. Nous pouvons, par exemple, réaliser des simulations non stationnaires (section 4.5.3) ou nous pouvons synthétiser des images ayant un histogramme différent de celui de l'image de référence (section 5.8.4).

Dans la nouvelle méthode PSM, nous avons proposé deux méthodes pour contrôler la moyenne locale (décrites à la section 4.4). La première, appelée *controlled histogram under stationary assumption* ou CHUSA, consiste à écrire les probabilités de transition $p_{j,k}$ en posant deux hypothèses :

1. le collage est toujours parfait, c'est-à-dire qu'il est toujours possible d'observer l'état de départ $\boldsymbol{U}(n)$ dans l'image de référence ;
2. la distribution des états collés est stationnaire et respecte $\boldsymbol{S}(n) \overset{\mathrm{d}}{=} \boldsymbol{S}_t$ à chaque pas n de la simulation où le symbole $\overset{\mathrm{d}}{=}$ signifie une égalité en distribution de probabilité et où $\boldsymbol{S}_t$ est le vecteur aléatoire visé ou *target random vector*.

Ce vecteur aléatoire $\boldsymbol{S}_t$ possède deux propriétés importantes :

1. il satisfait la contrainte imposée à l'histogramme de la moyenne locale (par exemple avoir beaucoup de chance d'être foncé et peu de chance d'être clair) ;
2. si la contrainte imposée à l'histogramme de la moyenne locale est de reproduire l'image de référence, alors $\boldsymbol{S}_t \overset{\mathrm{d}}{=} \boldsymbol{S}_r$, où $\boldsymbol{S}_r$ est un état $\boldsymbol{s}$ choisi au hasard et de façon équiprobable dans l'image de référence.

Bien que la méthode de contrôle CHUSA ait une certaine efficacité pour les champs aléatoires homogènes, elle ne permet pas de reproduire des histogrammes statistiquement compatibles avec l'objectif. Nous avons donc modulé les probabilités $p_{j,k}$ à l'aide de poids $\omega(i)$ afin d'aider les classes déficientes à respecter l'histogramme visé. Cette méthode utilisant des poids ajustés

est appelée *adaptive-CHUSA*.

Une autre amélioration importante apportée à la méthode PSM par rapport à la version originale est l'adaptation aux données conditionnantes. À la section 4.2.3, nous décrivons une approche qui permet de considérer les données conditionnantes se trouvant le long du chemin de simulation. Cette méthode, inspirée de la méthode unilatérale de Parra et Ortiz (2011), utilise une région étendue de l'état $\boldsymbol{S}(n)$ afin de limiter le choix des morceaux à coller à des morceaux compatibles avec les données conditionnantes futures.

Malgré cette adaptation aux données conditionnantes, nous avons constaté que la méthode unilatérale présente un biais causé par la direction de la simulation. Ce biais est facilement observable par l'anisotropie créée autour des données conditionnantes (voir figure 4.19). Pour corriger ce problème, nous avons développé une méthode alternative de simulation qui n'est pas unilatérale.

2.3 La méthode CPMS

Le chapitre 5 décrit la deuxième méthode de simulation par rapiéçage appelée *corrective pattern-matching simulation* (CPMS). Dans cette méthode, le chemin suivi n'est plus unilatéral mais est plutôt guidé par une mesure d'*erreur locale* notée $e(j)$, où j est un indice associé à une position dans l'image simulée et où l'erreur locale est mesurée dans une région de taille $L \times L$. À chaque pas de la simulation, un emplacement ayant une grande erreur locale est choisi pour être corrigé, où une correction consiste à coller un morceau de l'image de référence de taille $L \times L$ afin d'améliorer l'image simulée. Le choix de l'emplacement à corriger se fait au hasard avec une probabilité proportionnelle à l'erreur locale $e(j)$. L'erreur locale est une combinaison linéaire de trois erreurs locales :

1. l'erreur locale d'agencement $e_{\mathrm{pc}}(j)$ (*pattern conformity*) qui mesure la conformité des motifs observés localement avec les motifs observés dans l'image de référence ;
2. l'erreur locale de conformité aux données conditionnantes $e_{\mathrm{hd}}(j)$ (*hard data*) qui mesure le degré de respect des données conditionnantes ;
3. l'erreur locale statistique $e_{\mathrm{stat}}(j)$ qui pénalise les emplacements où un état appartenant à une classe de l'histogramme en surplus est observé.

Une fois qu'un emplacement à été sélectionné pour correction, l'étape critique de la simulation est le choix du morceau à coller pour améliorer l'image simulée. Ce choix est guidé par trois objectifs globaux à atteindre :

1. reproduire les motifs de l'image de référence ;
2. satisfaire les données conditionnantes ;
3. s'assurer que l'histogramme de la moyenne locale de l'image simulée soit statistiquement compatible avec l'histogramme défini par l'usager.

Le morceau à coller est choisi pour qu'il apporte une petite amélioration locale de sorte que, globalement, l'image simulée tende vers un optimum global.

Bien que la méthode CPMS ne présente pas le biais directionnel de la méthode PSM et qu'elle permette de bien reproduire les structures à l'échelle L, elle ne permet pas de reproduire les structures d'échelles plus petites ou plus grandes.

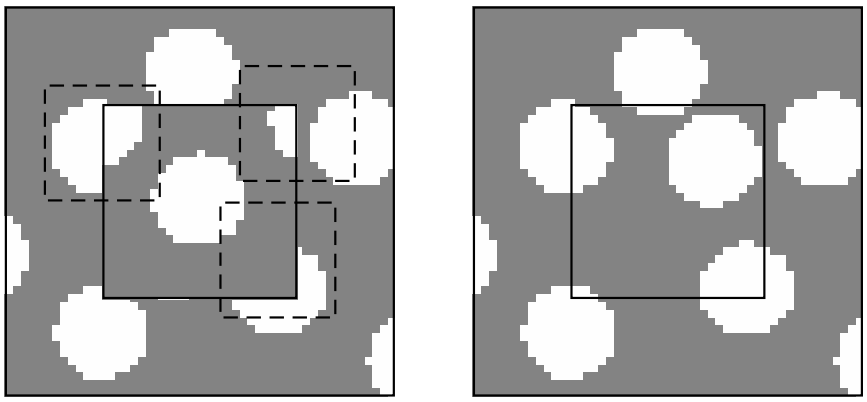
2.4 L'approche multi-échelle

Le chapitre 6 propose une approche multi-échelle utilisant notre méthode CPMS. Cette approche consiste simplement en une succession de plusieurs simulations CPMS utilisant des échelles L de plus en plus petites. Nous proposons également de fixer certaines valeurs de pixels comme des données conditionnantes après chaque simulation à une échelle donnée. La simulation peut ainsi garder certaines informations structurelles laissées par les échelles plus grossières.

CHAPITRE 3

CHOIX DE LA MÉTRIQUE

Dans une simulation par rapiéçage de motifs, nous procédons en prélevant des morceaux de l'image de référence et en les collant dans l'image simulée. Ce collage peut ajouter de nouvelles données ou modifier les données existantes. Considérons le cas où nous collons un morceau d'image sur des données déjà simulées. Il va alors se créer des discontinuités au pourtour du morceau collé. Par exemple, à la figure 3.1, le morceau d'image ne semble pas être un bon choix, car il ne s'assemble pas convenablement avec les données existantes. Nous disons alors que le collage crée des *erreurs d'agencement*. Ces erreurs peuvent être repérées visuellement et être mises en évidence. Par exemple, à la figure 3.1a, les encadrés en traits tiretés indiquent les régions où les disques sont mal formés.

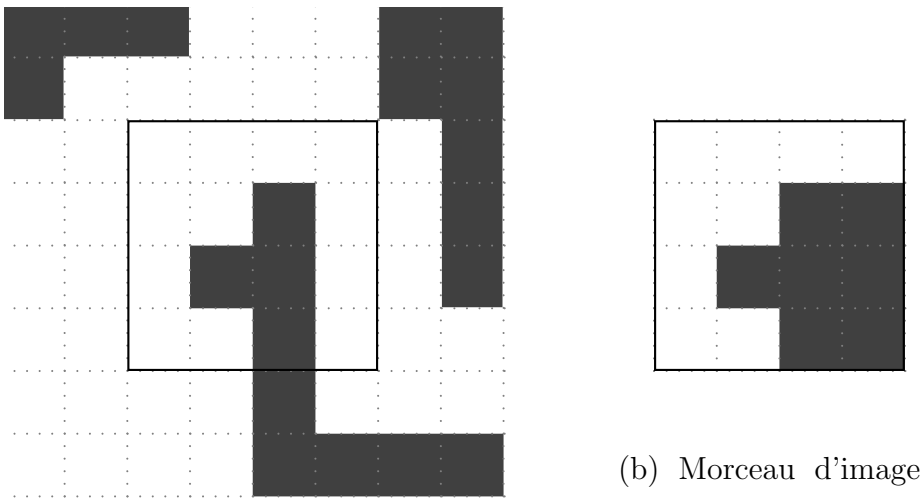


(a) Mauvais collage (b) Bon collage

Figure 3.1 Collage d'un morceau d'image (carré au centre).

Afin de définir cette erreur de manière quantitative, nous utilisons l'*erreur locale d'agencement* ou, en anglais, *local pattern-conformity error*. Nous définissons l'erreur locale d'agencement comme la plus petite distance entre le motif retrouvé dans une boîte de taille $L \times L$ et un motif retrouvé dans une boîte de même taille dans l'image de référence (la définition précise de l'erreur d'agencement est donnée au chapitre 5). Par exemple, l'erreur locale d'agencement

mesurée à la figure 3.2 est de $^3/_{16}$, car 3 pixels sur 16 diffèrent entre le morceau sélectionné et le morceau le plus proche dans l'image de référence.



(a) Boîte de taille 4×4 dans l'image simulée

(b) Morceau d'image le plus proche dans l'image de référence

Figure 3.2 Erreur locale d'agencement pour une boîte de taille 4×4. Ici, l'erreur est de $^3/_{16}$.

Étant donné que les erreurs d'agencement surviennent au pourtour du morceau collé, il est judicieux de s'assurer que le morceau collé s'agence aussi bien que possible au voisinage de ce pourtour. Pour considérer ce voisinage, nous devons considérer un plus grand morceau que le morceau collé. Par exemple, à la figure 3.1, le collage du morceau central de taille $L \times L$ se fait en fonction de tous les pixels du cadre de taille $2L \times 2L$. Cette méthode de collage est celle que nous utiliserons au chapitre 5. En reprenant le formalisme des sections précédentes, l'état $\boldsymbol{U}$ représente l'ensemble des valeurs des pixels dans le grand cadre et l'état $\boldsymbol{S}$ représente l'ensemble des valeurs des pixels du morceau collé (petit cadre central). La distance entre un état observé $\boldsymbol{u}_j$ et un état $\boldsymbol{u}$ de l'image de référence est calculée avec la métrique pondérée de l'équation (1.15).

La figure 3.3 illustre différents types de distribution des poids w_ℓ. La première distribution donne un poids constant aux pixels de la partie centrale. Avec cette distribution, le morceau collé est simplement celui qui minimise l'erreur locale d'agencement. Cette distribution constante des poids à l'emplacement du collage revient à utiliser la distance de Manhattan

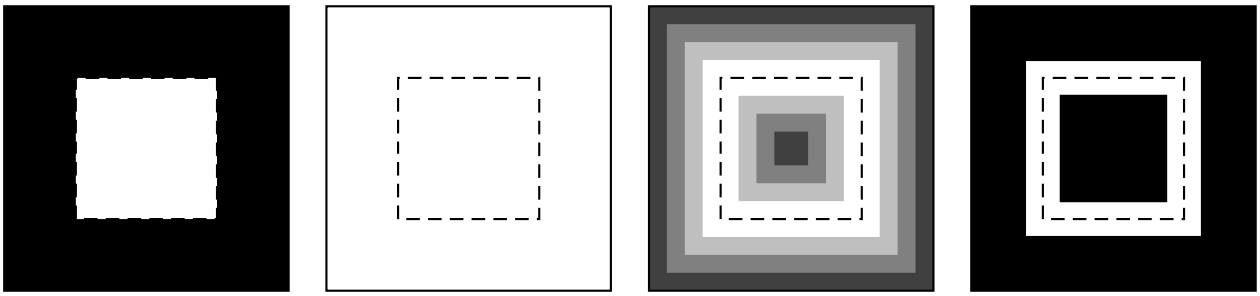


Figure 3.3 Quatre distributions spatiales de poids attribués aux pixels. Les tons de gris allant de noir à blanc correspondent à des valeurs allant de 0 à c où c est une constante de normalisation choisie telle que $\sum_{p=1}^{N_{\boldsymbol{u}}} w_{\boldsymbol{u}}(p) = 1$. La boîte intérieure en traits tiretés correspond à l'emplacement des nouvelles données (le morceau collé $\boldsymbol{S}$) et la boîte extérieure en traits pleins correspond au voisinage pris en compte pour le collage (l'état $\boldsymbol{U}$).

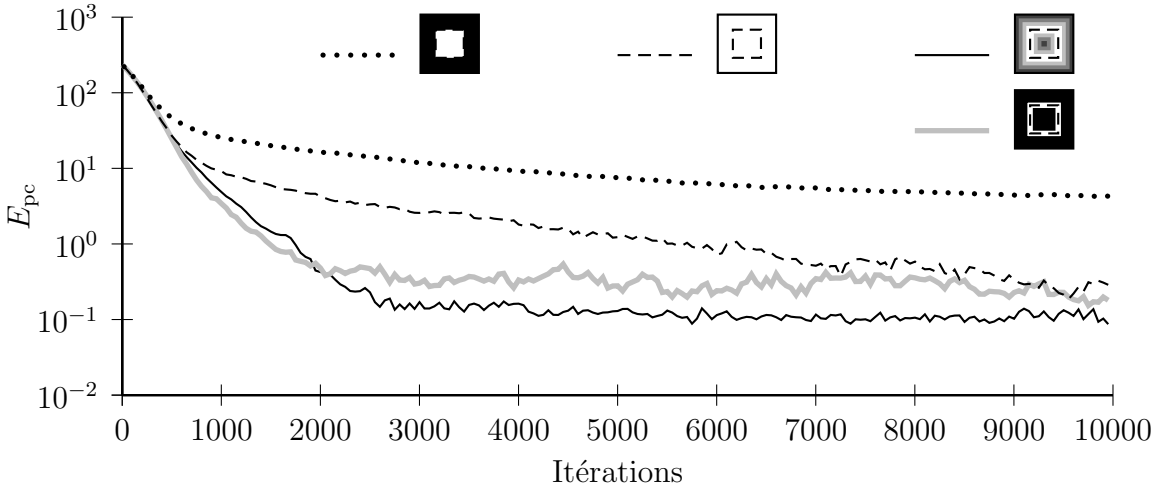


Figure 3.4 Évolution de l'erreur d'agencement E_{pc} pour les différentes distributions de poids illustrées à la figure 3.3. Les graphes sont obtenus à l'aide de simulations d'images de disques (voir sections 5.8.3).

comme dans la méthode *simpat* d'Arpat et Caers. La deuxième distribution donne un poids constant sur tout le voisinage considéré. Il s'agit donc d'une façon simple de considérer le voisinage pour assurer la continuité en dehors du cadre immédiat de la région de collage. La troisième distribution considère également tout le voisinage, mais concentre les poids sur le pourtour du cadre où se fait le collage. Dans cette distribution, les poids décroissent linéairement en s'éloignant du pourtour. Finalement, la quatrième distribution considérée ne donne d'importance qu'aux poids adjacents au pourtour.

Pour tester quelle distribution de poids est la plus appropriée, nous utilisons la méthode de simulation décrite en détail au chapitre 5. Dans cette méthode, l'image simulée est initialisée avec un bruit blanc, c'est-à-dire que chaque pixel prend la valeur 0 ou 1 avec une probabilité de $1/2$. Puis nous collons de manière itérative des morceaux de l'image de référence afin d'éliminer les erreurs d'agencement. À la figure 3.4, nous illustrons l'évolution de l'erreur globale d'agencement pour les quatre différentes distributions, où l'erreur globale d'agencement est une somme d'erreurs locales d'agencement. Ici, les erreurs locales d'agencement sont mesurées en prenant un carré de la taille des morceaux collés. Pour atténuer le bruit, les graphes de la figure 3.4 sont obtenus en prenant la moyenne de 10 simulations.

La figure 3.4 permet d'observer que les métriques donnant plus d'importance au pourtour sont celles qui minimisent le mieux l'erreur d'agencement. Bien que, selon le graphe, la quatrième distribution semble donner de bons résultats, nous observons en pratique que la simulation est plus instable avec cette distribution de poids, c'est-à-dire que la simulation est constamment en train d'ajouter et d'effacer des erreurs d'agencement. Pour cette raison le plateau moyen atteint est un peu plus élevé. Dans nos simulations, nous avons donc choisi la troisième distribution de poids.

CHAPITRE 4

ARTICLE 1: A NEW PATCHWORK SIMULATION METHOD WITH CONTROL OF THE LOCAL-MEAN HISTOGRAM

Corentin Faucher

École Polytechnique de Montréal, Mathématiques et génie industriel, C.P. 6079, Succursale Centre-ville, Montréal (Québec), Canada, H3C-3A7.

e-mail: corentin.faucher@polymtl.ca

Antoine Saucier

École Polytechnique de Montréal, Mathématiques et génie industriel, C.P. 6079, Succursale Centre-ville, Montréal (Québec), Canada, H3C-3A7.

Tel.: (1) 514-340-4711 ext. 4516

Fax: (1) 514-340-4463

e-mail: antoine.saucier@polymtl.ca

Denis Marcotte

École Polytechnique de Montréal, Génies civil, géologique et des mines, C.P. 6079, Succursale Centre-ville, Montréal (Québec), Canada, H3C-3A7.

e-mail: denis.marcotte@polymtl.ca

Publié en ligne le 9 juin 2012 chez Springer dans la revue *Stochastic Environmental Research and Risk Assessment* (SERRA).

DOI: 10.1007/s00477-012-0586-9.

Abstract

We present a new stochastic simulation method that builds two-dimensional images by assembling together square image pieces called blocks. The blocks are taken from a reference image. Our method, called patchwork simulation method (PSM), enforces pattern

continuity in the image. Moreover, PSM allows to control the image local-mean histogram. This histogram bin-frequencies can be set to user-defined target values that may differ from the reference image local-mean histogram. This flexibility enhances the PSM generality by enlarging the set of all possible simulations. The local-mean histogram control is achieved by adjusting suitably the transition probabilities that associate a new block to an existing neighborhood in the partly simulated image. For several types of synthetic images and one polymer blend image, we show that PSM reproduces faithfully the reference image visual appearance (i.e. patterns are correctly shaped) and that simulated images are statistically compatible with the target local-mean histogram. Moreover, we show that our method has the ability to produce simulations that respect conditional hard data as well as a target local-mean histogram.

Keywords simulations with patterns •reference image •histogram control •pattern similarity •multiple-point statistics •geostatistics •unilateral path

4.1 Introduction

The estimation of natural resources distributions and the assessment of their spatial uncertainty are problems that have been approached via geostatistical interpolation techniques (e.g. Deutsch and Journel, 1998; Saito, McKenna, Zimmerman and Coburn, 2005) and stochastic simulations (e.g. Emery, 2004; Zhang and Li, 2008; Arpat and Caers, 2007; Wu, Boucher and Zhang, 2008), which are increasingly used in geosciences.

Stochastic sequential simulations are used in geostatistics to model random fields that represent the spatial distribution of geological formations such as oil reservoirs or ore deposits. Simulation methods based on two-point statistics, e.g. sequential gaussian simulation or sequential indicator simulation (Deutsch and Journel, 1998), allow to reproduce efficiently the histogram and variogram of a given random field. However, reproducing two-point statistics is not sufficient to reproduce faithfully the complex patterns found in a random field. Reproducing such patterns can be important for some applications. For example, reproducing correctly the connectivity properties of a permeability field is essential if one is interested in subsurface fluid flow, since fluids tend to flow through permeable channels. To reproduce more accurately complex geometrical properties, simulation methods based on *multipoint*

statistics have recently been proposed.

Guardiano and Srivastava (1993) first proposed a multipoint sequential simulation algorithm in which the conditional probabilities are estimated from a reference image. It was found that this algorithm was quite demanding computationally because the whole reference image must be scanned at each simulation step. An improved version of this algorithm, called *snesim*, was proposed by Strebelle (2000; 2002; 2003). In this algorithm, the conditional probabilities are still estimated from a reference image but they are stored in a search tree, which allows reducing the computing load. To take into account the random field multiscale properties and to limit the size of the neighborhood in which the search is performed, Strebelle used a structure called multiple-grid for the construction of simulated images.

Parra and Ortiz (2011) proposed an adaptation of the texture synthesis algorithm of Wei and Levoy (2000) to geostatistical simulation. Their simulation proceeds by visiting single nodes in a unilateral way (point by point and line by line), like a Markov random field. Conditioning is taken into account by adding a non-causal region to the search neighborhood, i.e. the unilateral simulation looks ahead to ensure that simulated values are consistent with forthcoming hard data.

Another family of multipoint sequential simulation methods, called patchwork simulation methods (PSM) in the following, builds the simulated image by assembling *patterns* together. Patterns are image pieces taken from a reference image. The simulation proceeds like a jigsaw puzzle, i.e. by sticking image pieces to the simulated image. The advantage of PSM is that patterns inside each image piece are correct to start with, since they are taken directly from the reference image. This advantage also holds for the simulated image if the assembly process preserves the image pieces integrity and if the image pieces are assembled correctly, e.g. without creating continuity errors at the boundary between two pieces. Arpat and Caers (2007) proposed a *conditional simulation with patterns* that proceeds by sticking a pattern (a cubic image-piece) at each visited location along a random path. This pattern is chosen so that it fits with previously simulated data and conditioning hard data.

Ouassini, Saucier and Marcotte (2006; 2008) proposed a PSM that proceeds sequentially by assembling together square-shaped images pieces called blocks. The simulation is unilateral in the sense that blocks are assembled together and in a pre-determined order, i.e. from

left to right and top to bottom. Given an existing local neighbor block, this method uses transition probabilities to determine which block comes next. The PSM has a resemblance with a Markov mesh random field (Pickard, 1980; Razlighi *et al.*, 2009), for which the value of a pixel P is a random variable that depends only on the pixels which are in the immediate neighborhood of P. The main difference between a Markov random field and a random field obtained via the PSM is that the latter uses multidimensional states that represent patterns, instead of unidimensional real numbers that represent single-pixel values.

The PSM was shown to produce very good results in the sense that simulated images reproduce faithfully the reference-image patterns at small scales, i.e. for scales smaller than the block size. The excellent visual appearance of the simulations demonstrated clearly that blocks are correctly assembled together to form larger patterns. However, it was also observed that this method does not control the pixel values probability distribution, which may result for example in a simulated image having an average pixel value significantly lower, or higher, than the reference image.

The ability to reproduce one-point statistics such as the mean is important in geostatistical simulations. For instance, the facies proportions in an oil reservoir may be known approximately, e.g. via seismic data. In this context, simulation of this reservoir should reproduce both the facies proportions and the patterns observed in a reference image. In this perspective, Strebelle (2000) proposed a *servo system* to get a *snesim* simulation to match target facies proportions. More recently, an improved version of this servo system was proposed by Tuanfeng, Stein and McCormick (2008) to deal efficiently with more than two facies. In another version of the *snesim* simulation (Boucher and Stright, 2008), the simulation of fine scales is constrained by coarse scale measurements.

Recently introduced simulation methods that exploit training images and multipoint statistics (e.g. Arpat and Caers, 2007; Strebelle, 2000; Wu, Boucher and Zhang, 2008; Zhang, Switzer and Journel, 2006) have one common weakness: they lack a fine and direct control on the image statistics. For example, the conditional simulation with patterns (simpat) of Arpat and Caers (2007) does not introduce any stochastic modeling, nor any assurance that the reference image statistics are actually reproduced. Also, the servo system introduced by Strebelle (2000) in the snesim simulations allows to control facies proportion, i.e. the average

pixel value for a black and white image, which is a statistical control restricted to the image mean-value. Similarly, the method filtersim (Wu, Boucher and Zhang, 2008; Zhang, Switzer and Journel, 2006), which proposes a simulation based on suitably classified patterns, does not provide any direct control of the image statistics.

In this paper, we propose an improved version of the unilateral patchwork simulation method proposed by Ouassini, Saucier, Marcotte and Favis (2006; 2008). This new method has two main advantages with respect to the aforementioned methods. Firstly, we have a direct control of the image statistics, i.e. a control on the local-mean probability distribution function (PDF). It is emphasized that this control on the local-mean at the scale L of a block also implies a control of multiple-point statistics at scales smaller than L. Secondly, we enforce rigorously the continuity of patterns in the image, which results in very smoothly constructed patterns.

Our control on the *local-mean histogram* is achieved by suitably modifying the transition probabilities and by partitioning the set of all the image-pieces found in the reference image according to their local-mean. Our new method allows to simulate images that have the same histogram as the reference image, but also allows to simulate images that have a histogram that differs from the reference image histogram, while preserving a similar visual appearance. In addition to controlling the local-mean histogram, our method also allows to take into account conditioning hard data.

4.2 Overview and preliminary definitions

4.2.1 The patchwork simulation method

We construct images using data from a reference image that is representative of the random field to simulate. The simulation proceeds sequentially by patching square-shaped image pieces into a square-shaped region called *patching square*. The location of the patching square changes at each simulation step. Square-shaped image pieces will be called *blocks* (Figure 4.1a). Both blocks and patching squares have the same size $L = 4i$, where $i \geq 1$ is a integer. Starting from a patching square that contains an L-shaped image piece (Figure 4.1c), patching consists in sticking the lower-right part of a suitably chosen block into the

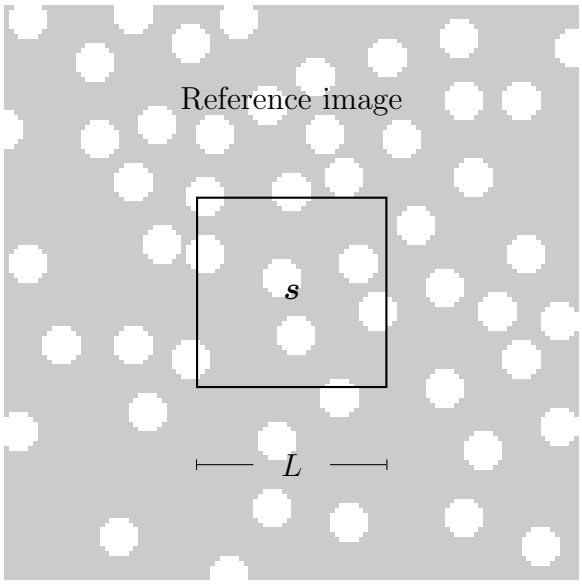
lower-right part of the patching square (Figure 4.1d), thus expanding the image.

The patching square is initially located at the upper-left corner of the simulated image and moves from left to right and top to bottom as the simulation proceeds. At each step, a new block of the reference image is patched to the simulated image. The block patched into the patching square is chosen conditionally to the data already simulated. Part of this conditioning ensures that the chosen block resembles previously simulated data in the patching square, i.e. data in the L-shaped region illustrated in Figure 4.1c. This resemblance ensures the continuity of patterns in the simulated image.

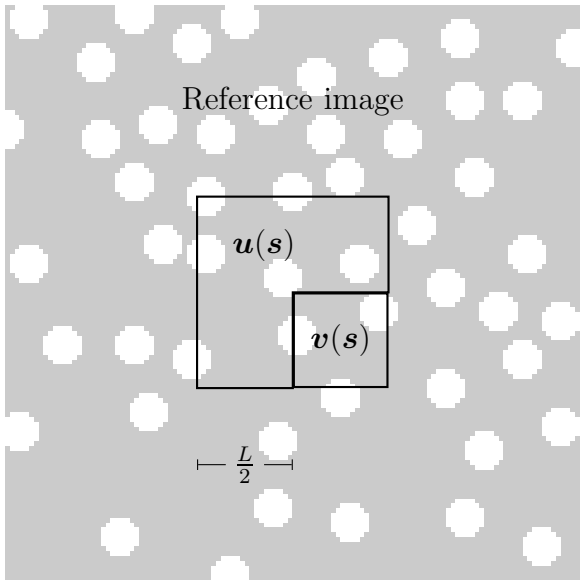
In the following, a *state* is defined as a vector whose components are the pixel values found in a given image region, called the state *support*. We use four different types of states that are distinguished by the shape of their support (Figure 4.1). The following notations are important because they will be used extensively in this paper. States associated to a square support of size L (Figure 4.1a) will be called *block-states* and will be denoted by $\boldsymbol{s}$ or $\boldsymbol{s}_k$, where k is an index. States associated to L-shaped regions of size L (Figure 4.1c) will be called *L-states* and will be denoted by $\boldsymbol{u}$ or $\boldsymbol{u}_j$, where j is an index. The L-shaped part of a block-state $\boldsymbol{s}$, which is a L-state, will be denoted by $\boldsymbol{u}(\boldsymbol{s})$ (Figure 4.1b). The lower-right square part of size $L/2$ of a block state $\boldsymbol{s}$, denoted by $\boldsymbol{v}(\boldsymbol{s})$ (Figure 4.1b), will be called a *quarter-state*. The lower-right square part of size $3L/4$ of a block state $\boldsymbol{s}$, denoted by $\boldsymbol{a}(\boldsymbol{s})$ (Figure 4.1d), will be called a *patch-state*.

It will be assumed that the set of all possible pixel values is discrete and finite. It follows that the set of all the states of a given type (e.g. block-states) found in a reference image is also discrete and finite. We define $\mathcal{S}_r = \{\boldsymbol{s}_1, \boldsymbol{s}_2, ...\}$ to be the set of all the distinct block-states $\boldsymbol{s}$ found in the reference image ($\mathcal{S}_r$ is finite and countable). If the reference image is statistically isotropic, then we also include in $\mathcal{S}_r$ the states found in the four 90° rotations of the reference image and their mirror symmetries. In this case, each reference image generates via symmetries seven additional reference images. Similarly, we define $\mathcal{U}_r = \{\boldsymbol{u}_1, \boldsymbol{u}_2, ...\}$ to be the set of all the distinct L-states $\boldsymbol{u}$ found in the reference image, including states obtained via symmetries.

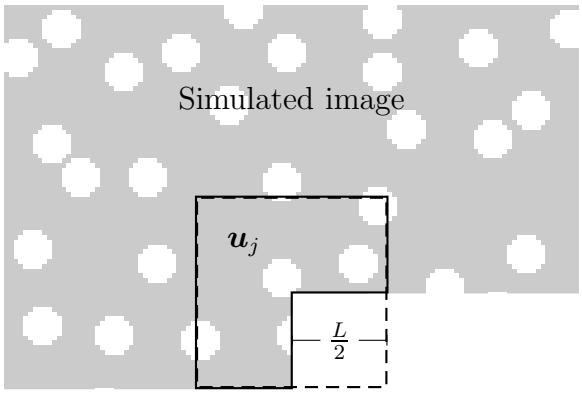
The reference image naturally defines a probability for each state. Assuming that all blocks are equiprobable, choosing a block randomly in the reference image defines a reference



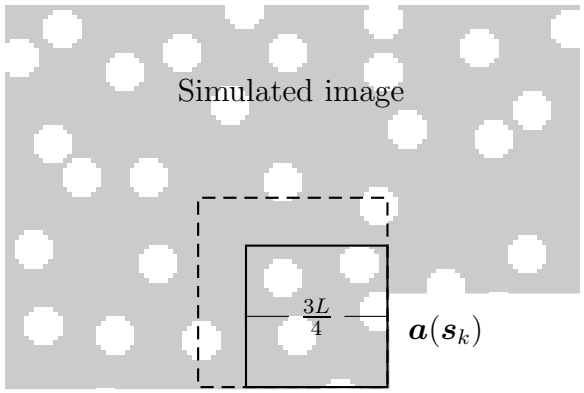
(a) A block-state $\boldsymbol{s}$



(b) L-state $\boldsymbol{u}(\boldsymbol{s})$ and quarter-state $\boldsymbol{v}(\boldsymbol{s})$



(c) L-state $\boldsymbol{u}_j$ in the patching square



(d) Patching of $\boldsymbol{a}(\boldsymbol{s}_k)$ (solid square) to the patching square

Figure 4.1 a) A block-state $\boldsymbol{s}$ is an array that contains all the pixel values of an image square block of size L (solid square). b) The L-state $\boldsymbol{u}(\boldsymbol{s})$ is the L-shaped part of the block-state $\boldsymbol{s}$. The quarter-state $\boldsymbol{v}(\boldsymbol{s})$ (small solid square) is the lower-right part of the block-state $\boldsymbol{s}$. c) A partly simulated image and a L-state $\boldsymbol{u}_j$ in the patching square (dashed). d) Patching process: the block-state $\boldsymbol{s}_k$ is chosen in the reference image so that its L-shaped part $\boldsymbol{u}(\boldsymbol{s}_k)$ is close to $\boldsymbol{u}_j$ in (c). The lower-right part of $\boldsymbol{s}_k$, denoted by $\boldsymbol{a}(\boldsymbol{s}_k)$ (solid square), is patched to the partly simulated image, thus expanding the image size.

random vector denoted by $\boldsymbol{S}_r$. The probability of picking a block-state $\boldsymbol{s}_k$ in the reference

image is given by

$$\mathrm{P}\left[\boldsymbol{S}_r = \boldsymbol{s}_k\right] = \frac{c(\boldsymbol{s}_k)}{\sum\limits_{\boldsymbol{s}_i \in \mathcal{S}_r} c(\boldsymbol{s}_i)}, \tag{4.1}$$

where $c(\boldsymbol{s}_k)$ denotes the number of occurrences of $\boldsymbol{s}_k$ in the reference image.

4.2.2 Transition probabilities and pattern continuity

Let us now describe the patchwork simulation method more precisely. At the n^{th} simulation step, a block-state $\boldsymbol{S}(n) \in \mathcal{S}_r$ is chosen randomly to add data to the simulated image. The state $\boldsymbol{S}(n)$ is chosen conditionally to existing data. We assume that $\boldsymbol{S}(n)$ depends only on data that are contained in the patching square. This assumption is approximately valid if the block size L is larger or equal to the reference image correlation length [1]. As seen in Figure 4.1c, the conditioning data in the patching square is found in a L-shaped region containing a state denoted by $\boldsymbol{U}(n) \in \mathcal{U}_r$. The patchwork simulation is driven by the transition probabilities

$$p_{j,k} := \mathrm{P}\left[\boldsymbol{S}(n) = \boldsymbol{s}_k | \boldsymbol{U}(n) = \boldsymbol{u}_j\right], \tag{4.2}$$

which are assumed to be independent of n. The choice of these probabilities will be discussed in the sections 4.4.3 and 4.4.4. For the state $\boldsymbol{S}(n)$ chosen conditionally to $\boldsymbol{U}(n)$, patching consists in sticking $\boldsymbol{a}(\boldsymbol{S}(n))$ to the simulated image, where $\boldsymbol{a}(\boldsymbol{s}_k)$ denotes the substate of $\boldsymbol{s}_k$ associated to the block of size $\frac{3L}{4}$ at the lower-right of the block of $\boldsymbol{s}_k$ (Figure 4.1d). Note that unlike the PSM of Ouassini *et al.* (2006; 2008), in which the whole block-state $\boldsymbol{s}_k$ is superposed to the previous state $\boldsymbol{u}_j$, we use only a substate $\boldsymbol{a}(\boldsymbol{s}_k)$ of size $\frac{3L}{4}$ for the superposition. We found that this modification reduces significantly the number of pattern continuity errors.

The block-state $\boldsymbol{S}(n)$ is chosen so that the L-state $\boldsymbol{u}(\boldsymbol{S}(n))$ is similar or even identical to the L-state $\boldsymbol{U}(n)$ contained in the patching square. The quarter-state $\boldsymbol{v}(\boldsymbol{S}(n))$ is the part of the patch that makes the image grow in size at each step. To take into account the conditioning of $\boldsymbol{S}(n)$ by $\boldsymbol{U}(n)$, we limit the choice of $\boldsymbol{S}(n)$ to the set $A(\boldsymbol{U}(n))$, where $A(\boldsymbol{u}_j)$ denotes the set of all the block-states $\boldsymbol{s} \in \mathcal{S}_r$ such that the L-state $\boldsymbol{u}(\boldsymbol{s})$ is closest to $\boldsymbol{u}_j$. More

1. The correlation length is closely related to the variogram range.

precisely,

$$A(\boldsymbol{u}_j) := \underset{\boldsymbol{s}\in\mathcal{S}_r}{\operatorname{argmin}}\, d\left(\boldsymbol{u}(\boldsymbol{s}), \boldsymbol{u}_j\right), \tag{4.3}$$

where $d\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right)$ denotes the distance between the L-states $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$. Hence the set $A(\boldsymbol{u}_j)$ contains one or several block-states that are at an *equal distance* from $\boldsymbol{u}_j$. The distance function $d\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right)$ appearing in (4.3) is defined by

$$d\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) := \left\|\boldsymbol{u}_i - \boldsymbol{u}_j\right\|, \tag{4.4}$$

where the norm $\|\ldots\|$ is defined for any $\boldsymbol{u}_k \in \mathbb{R}^m$ by

$$\|\boldsymbol{u}_k\| = \sum_{i=1}^{m} w(i)|u_k(i)|. \tag{4.5}$$

The $w(i)$s in (4.5) are strictly positive weights. The spatial distribution of the weights $w(i)$ is chosen to favor patterns continuity in the simulated image. When patching new data into the simulated image, we may create pattern discontinuities at the border between existing data and added data (border of $\boldsymbol{a}(\boldsymbol{s}_k)$ in Figure 4.1c). To favor the selection of a block-state $\boldsymbol{s}_k$ that fits well at the border between new data and existing data, we give a larger weight to the pixels that are close to this border. In Figure 4.2, we illustrate the spatial distribution of the weights $w(i)$ for $L = 16$. The weights are set to their maximum value (four) for the pixels that are closest to the border (white strip), and to their minimum value (one) for the pixels that are on the edge of the L-shape region (darkest strip). The weights vary linearly between their maximum and minimum values.

One of the simplest way to define the transition probabilities (4.2) is to assume that all block-states $\boldsymbol{s}_k \in A(\boldsymbol{u}_j)$ are equiprobable, which leads to

$$p_{j,k} = \begin{cases} \dfrac{1}{\operatorname{card}\left(A(\boldsymbol{u}_j)\right)} & \text{if } \boldsymbol{s}_k \in A(\boldsymbol{u}_j), \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

Simulations performed using (4.6) will be called *equiprobable nearest neighbor* (ENN) simulations. These simulations are the simplest to perform, they insure a good continuity of

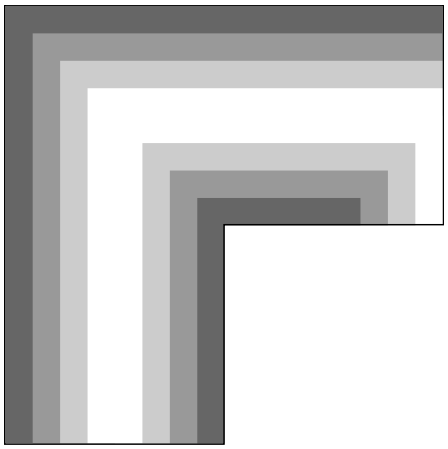


Figure 4.2 Spatial distribution of the weights $w(i)$ with $L = 16$ pixels. Dark gray and white correspond to the weight values one and four respectively.

the image patterns, but they do not provide any statistical control on the simulated image. Nevertheless, we will use ENN simulations because they provide a simple comparison point for the more sophisticated statistical control methods that will be proposed in section 4.4.

4.2.3 Conditioning to hard data

Hard data are a set of predetermined pixels values. To take hard data into account, we use an approach inspired of the work of Parra and Ortiz (2011), who proposed a search neighborhood that includes a non-causal region for the forthcoming hard data. At each simulation step n, we consider only the hard data that is contained in a region near the patching-square. We consider either the support of the patch-state $\boldsymbol{a}(\boldsymbol{S}(n))$ (Figure 4.3a), called the *short-range conditioning region*, or the larger square region of size $3L/4 + L/2$ displayed in Figure 4.3a, called the *extended-range conditioning region*. The hard data contained in the conditioning region will be called the *local hard data*. If no local hard data are found in the conditioning region, then the block-state $\boldsymbol{S}(n)$ is picked in the set $A(\boldsymbol{U}(n))$. However, if the conditioning region does contain hard data, then $\boldsymbol{S}(n)$ must be picked in a different set, as defined below.

For a short-range hard data conditioning, $\boldsymbol{S}(n)$ is picked in the subset $\mathcal{S}_{r,\,n} \subset \mathcal{S}_r$ that contains only states that honor the short-range hard data at step n. To respect both hard

data and pattern continuity, we pick $\boldsymbol{S}(n)$ in the set $A_n(\boldsymbol{U}(n))$ defined as

$$A_n(\boldsymbol{U}(n)) := \underset{\boldsymbol{s}\in\mathcal{S}_{r,n}}{\operatorname{argmin}}\ d(\boldsymbol{u}(\boldsymbol{s}), \boldsymbol{U}(n)). \tag{4.7}$$

Note that the set $\mathcal{S}_{r,n}$ may be empty if the conditioning hard-data is not compatible with the reference image. In this case, a natural way to solve this problem is to use a reference image that contains a greater variety of distinct patterns.

The extended-range hard data conditioning makes use of *extended block-states* of size $3L/2$, denoted by $\boldsymbol{s}^*$. An extended block-state $\boldsymbol{s}^*$ is defined as a state associated to a support composed of the union of two squares, one of size L and another of size $3L/4+L/2$, positioned with respect to each other as shown in Figure 4.3b. We chose the support shape of extended block-states so that it matches exactly the shape of the region composed of the union of the patching-square and the extended-range conditioning region, as shown in Figure 4.3a. The set of all distinct states $\boldsymbol{s}^*$ found in the reference image is denoted by $\mathcal{S}_r^*$, and the upper-left L-shaped part of $\boldsymbol{s}^*$ is a L-state denoted by $\boldsymbol{u}^*(\boldsymbol{s}^*) \in \mathcal{U}_r$ (Figure 4.3b). The extended-range hard data conditioning is performed by picking the extended block-state $\boldsymbol{S}^*(n) \in \mathcal{S}_r^*$ in the subset $\mathcal{S}_{r,n}^* \subset \mathcal{S}_r^*$ that contains only states that honor the extended-range hard data at step n. To respect both hard data and pattern continuity, we pick $\boldsymbol{S}^*(n)$ in the set $A_n^*(\boldsymbol{U}(n))$ defined as

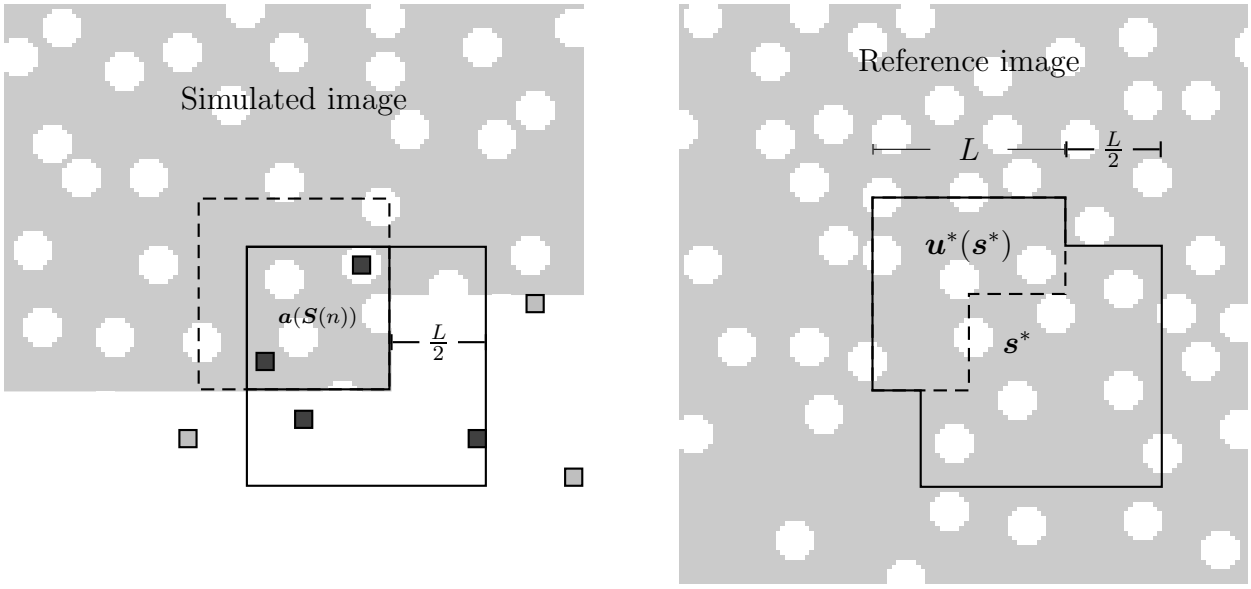
$$A_n^*(\boldsymbol{U}(n)) := \underset{\boldsymbol{s}^*\in\mathcal{S}_{r,n}^*}{\operatorname{argmin}}\ d(\boldsymbol{u}^*(\boldsymbol{s}^*), \boldsymbol{U}(n)). \tag{4.8}$$

4.3 Construction process

In this section, we describe the patchwork simulation construction method in detail.

4.3.1 Initialization

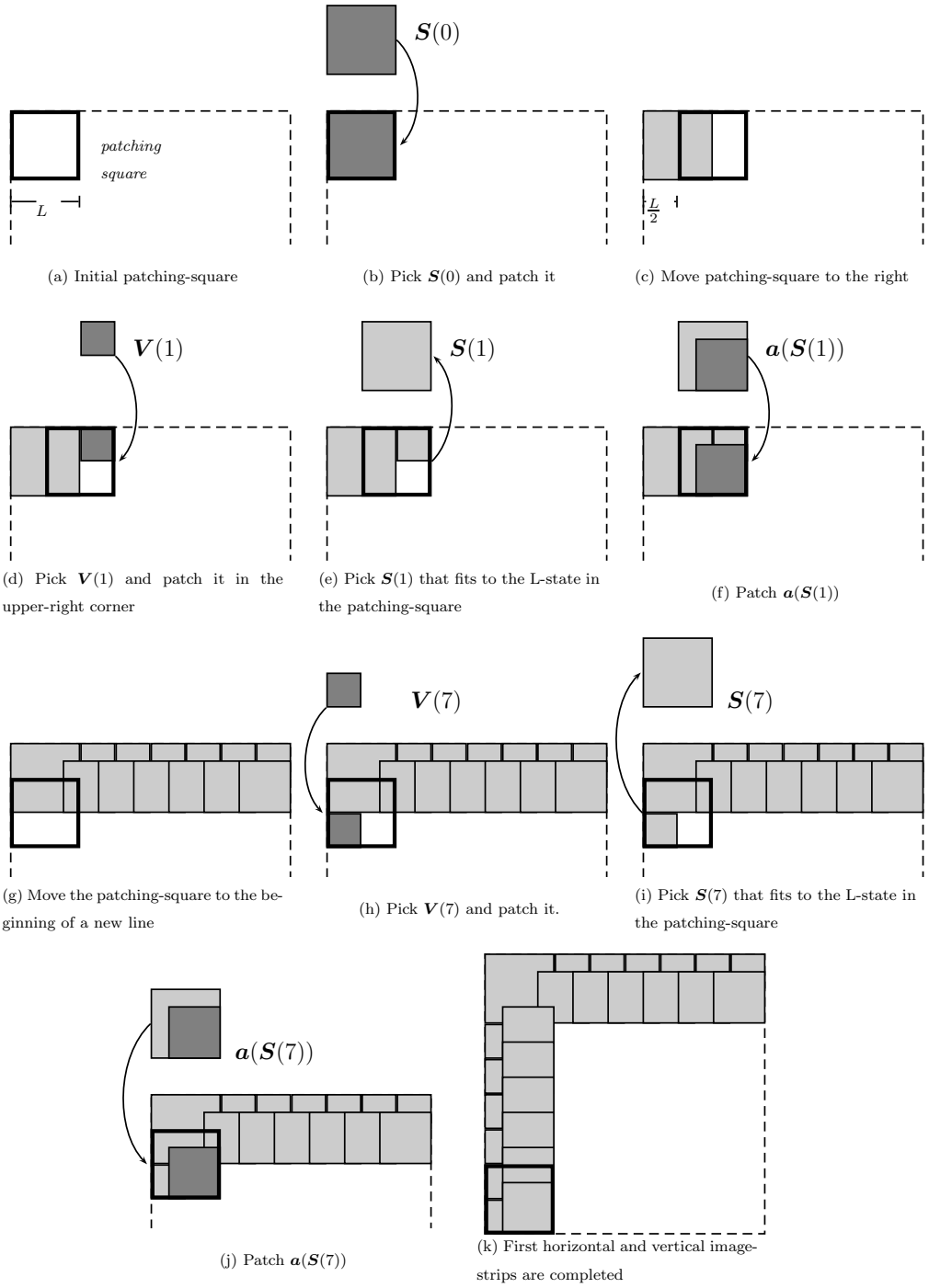
The first step of the simulation is to build a horizontal and a vertical image strip at the top and the left of the simulated image. This initialization step provides data on which we build the simulation. The patching square is initially located at the upper left corner of the simulated image (Figure 4.4a).



(a) Conditional regions

(b) The extended block-state $\boldsymbol{s}^*$

Figure 4.3 Conditioning to hard data: **a)** local hard data are depicted as small dark-gray squares. The patching-square is delimited by dashed line segments and coincides with the support of $\boldsymbol{S}(n)$. The square support of $\boldsymbol{a}(\boldsymbol{S}(n))$ is the short-range conditioning region. The larger square region of size $3L/4 + L/2$, which is delimited by solid line segments, is the extended-range conditioning region. The support of an extended block-state $\boldsymbol{s}^*$ has the same shape as the region composed of the union of the patching-square and the extended range conditioning region. **b)** The support of an extended block-state $\boldsymbol{s}^*$ in the reference image is the union of two squares. The L-state $\boldsymbol{u}^*(\boldsymbol{s}^*)$ is in the upper-left corner of the support of $\boldsymbol{s}^*$.



(a) Initial patching-square

(b) Pick $\boldsymbol{S}(0)$ and patch it

(c) Move patching-square to the right

(d) Pick $\boldsymbol{V}(1)$ and patch it in the upper-right corner

(e) Pick $\boldsymbol{S}(1)$ that fits to the L-state in the patching-square

(f) Patch $\boldsymbol{a}(\boldsymbol{S}(1))$

(g) Move the patching-square to the beginning of a new line

(h) Pick $\boldsymbol{V}(7)$ and patch it.

(i) Pick $\boldsymbol{S}(7)$ that fits to the L-state in the patching-square

(j) Patch $\boldsymbol{a}(\boldsymbol{S}(7))$

(k) First horizontal and vertical image-strips are completed

Figure 4.4 Initialization of the simulation. The *patching-square* is represented by a bold square. White, light-gray and dark-gray pixels represent undefined data, simulated data and newly added data respectively.

First, we choose randomly a block-state $\boldsymbol{S}(0) \overset{\mathrm{d}}{=} \boldsymbol{S}_r$, where the symbol $\overset{\mathrm{d}}{=}$ denotes an equality in probability distribution function [2], and we patch all its data into the patching square (Figure 4.4b). Next, we move the patching square to the right by $\frac{L}{2}$ pixels (Figure 4.4c). Next, we choose randomly a quarter-state $\boldsymbol{V}(1) \overset{\mathrm{d}}{=} \boldsymbol{v}(\boldsymbol{S}_r)$ independently of $\boldsymbol{S}(0)$ and put it in the upper-right corner of the patching square (Figure 4.4d). At this point, the patching square contains a L-state denoted by $\boldsymbol{U}(1)$. For this state $\boldsymbol{U}(1)$, we choose randomly a block-state $\boldsymbol{S}(1)$ using the transition probabilities (4.2) (Figure 4.4e). and we patch $\boldsymbol{S}(1)$ to the patching square (Figure 4.4f). To build a horizontal image strip, we repeat the operations illustrated in the Figures 4.4c-4.4f.

We emphasize that it is necessary to use the random quarter-states $\boldsymbol{V}(n)$ on the edges to obtain truly stochastic simulations. Indeed, simulations are strongly conditioned by their boundaries and consequently using fixed boundaries leads to very low variability simulations, which is not desirable. This initialization procedure can introduce assembly errors in the horizontal and vertical image strips because the quarter-state $\boldsymbol{V}(n)$ does not necessarily fit well with the previous state. These errors can be problematic because they can propagate throughout the entire simulated image (this is illustrated in Figure 4.7). We reduce the magnitude of these errors as follows. At each step of the initialization, we choose randomly 10 quarter-states $\boldsymbol{V}_i(n)$, $i = 1, \ldots, 10$ to which correspond a set $U := \{\boldsymbol{U}_i(n), i = 1, \ldots, 10\}$ of 10 candidate L-states $\boldsymbol{U}_i(n)$. From this set, we keep only one of the L-states $\boldsymbol{U}_k(n)$ that fits best to the reference image, i.e. $\boldsymbol{U}_k(n) \in \operatorname{argmin}_{\boldsymbol{U}_i(n) \in U,\, \boldsymbol{u}_j \in \mathcal{U}_r} d(\boldsymbol{U}_i(n), \boldsymbol{u}_j)$.

When the horizontal image strip is completed (Figure 4.4g), we use similar operations to construct a vertical image strip: starting with the patching square in the top-left corner, we move it down by $\frac{L}{2}$ pixels; we pick randomly a quarter-state $\boldsymbol{V}(7) \overset{\mathrm{d}}{=} \boldsymbol{v}(\boldsymbol{S}_r)$ and put it in the lower-left corner of the patching square (Figure 4.4h); we pick randomly a block-state $\boldsymbol{S}(7)$ (Figure 4.4i) using the transition probabilities (4.2), and patch it to the patching square (Figure 4.4j). We repeat these operations until the vertical strip is completed.

2. In this case, this corresponds to choosing the value of $\boldsymbol{S}(0)$ according to the probability (4.1).

4.3.2 Main simulation

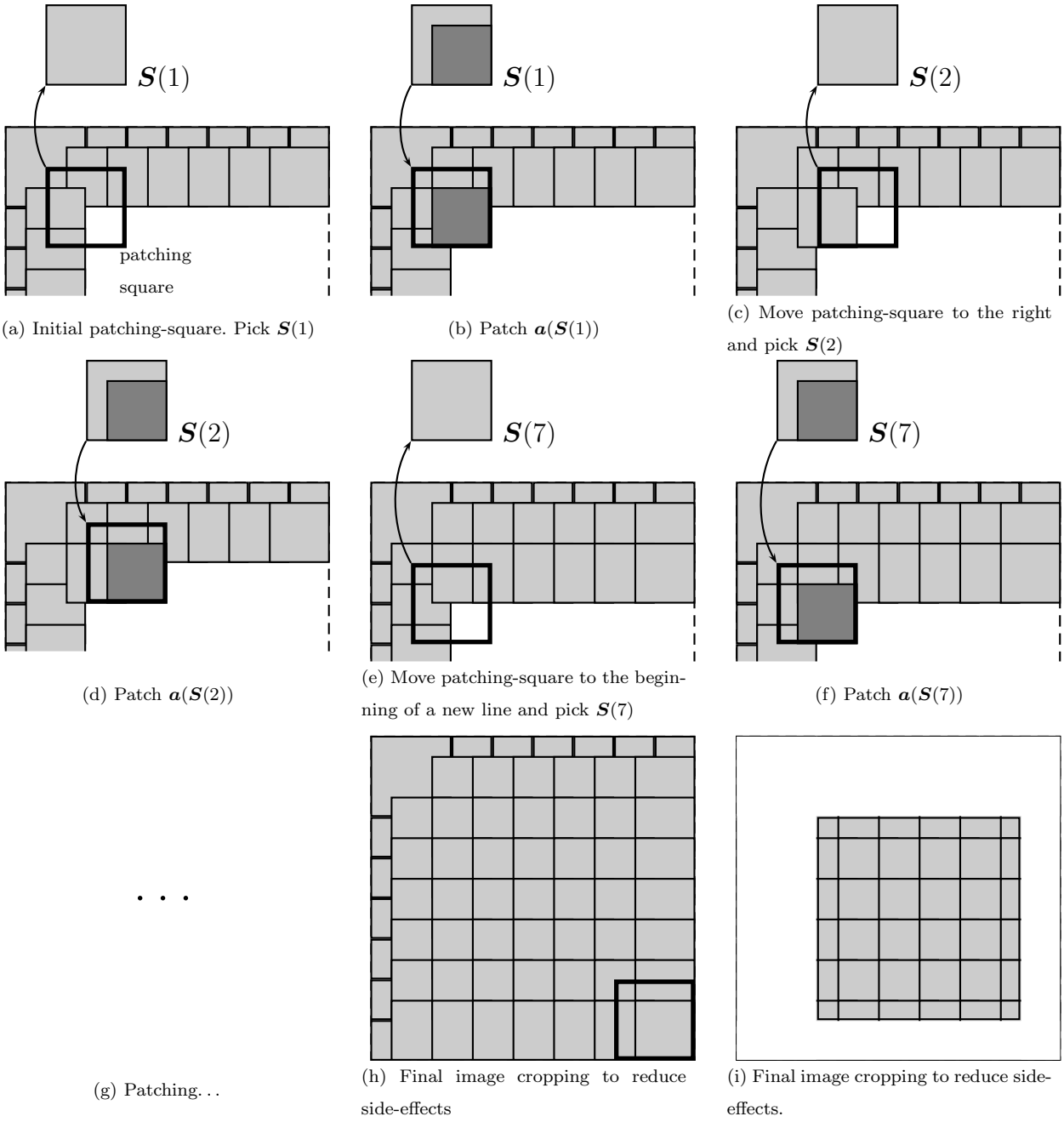
After the initialization, we start the main simulation process. The patching square is initially located in the upper-left corner at $\frac{L}{2}$ pixels from the left side and $\frac{L}{2}$ pixels from the upper side (Figure 4.5a). The patching square contains a L-state $\boldsymbol{U}(1)$. Firstly, we choose randomly a block-state $\boldsymbol{S}(1)$ using the transition probabilities (4.2). Secondly, we patch $\boldsymbol{S}(1)$ to the patching square (Figure 4.5b). Thirdly, we move the patching square to the right by $\frac{L}{2}$ pixels and repeat the patching operation (Figures 4.5c and 4.5d). Once the line is completed, we move the patching square back to the left at $\frac{L}{2}$ pixels from the left side and move it down by $\frac{L}{2}$ pixels to make a new line (Figures 4.5e and 4.5f). Finally, when the simulated image is completed, we crop the image to reduce side effects (Figures 4.5h and 4.5i). This is done by removing the two strips of size L built during initialization, as well as a horizontal and a vertical strip of width $\frac{L}{2}$ at the bottom and to the right of the image.

4.3.3 Choice of the block size

A critical parameter in our PSM is the block size L. We observed empirically in simulations that there is an optimal value of L for a given reference image, i.e. a value of L for which the simulations nicely reproduce the reference image visual aspect.

If L is too small, then patterns that have a diameter larger than L must be constructed via several patching steps. The repeated assembly of blocks to one another, without a large-scale statistical control, will introduce pattern distortions in the simulation. If L is too large, then patterns are well reproduced inside each patch-state $\boldsymbol{a}(\boldsymbol{s})$, but it becomes unlikely to find block-states $\boldsymbol{s} \in \mathcal{S}_r$ that fit well to the L-state $\boldsymbol{U}(n)$ because the set $\mathcal{S}_r$ contains fewer states for larger values of L and the number of possible values of the composite state $\boldsymbol{U}(n)$ increases fast as L increases.

The optimal value of L is closely related to the reference image correlation length. For reference images that have a characteristic pattern size, e.g. black disks thrown on a white background, this length corresponds approximately to the typical pattern size.



(a) Initial patching-square. Pick $\boldsymbol{S}(1)$

(b) Patch $\boldsymbol{a}(\boldsymbol{S}(1))$

(c) Move patching-square to the right and pick $\boldsymbol{S}(2)$

(d) Patch $\boldsymbol{a}(\boldsymbol{S}(2))$

(e) Move patching-square to the beginning of a new line and pick $\boldsymbol{S}(7)$

(f) Patch $\boldsymbol{a}(\boldsymbol{S}(7))$

(g) Patching...

(h) Final image cropping to reduce side-effects

(i) Final image cropping to reduce side-effects.

Figure 4.5 Main simulation.

4.3.4 Algorithmic considerations

An important improvement of this new version of the PSM has been the implementation of a tree structure for states storage. The most compute-intensive part of the construction process is the search at each step n of the block-states $\boldsymbol{s} \in A(\boldsymbol{U}(n))$ that best fit the existing

data $\boldsymbol{U}(n)$. To find the set $A(\boldsymbol{u})$, all the possible L-states $\boldsymbol{u}$ in the reference image must be scanned. In the simulations presented in this paper, we used images of size 200×200 and therefore there are $(200-(L-1))^2 \times 8 \simeq 274000$ states to scan at each simulation step.

To speed-up computations, we used a tree structure called *spill-tree* (Liu *et al.*, 2004) to store the L-states $\boldsymbol{u} \in \mathcal{U}_r$. To control the local-mean histogram (to be explained in section 4.4), one has to build one such tree for each local-mean class. This tree structure is highly efficient when dealing with high-dimensional clustered data such as the states found in images composed of disks or channels (sections 4.5.2 and 4.5.4). We found that this tree structure makes our simulation more than 300 times faster. For high entropy images, for which data are less clustered, a spill tree is long to build and inefficient. In this case, it is more suitable to use a metric-tree (Liu *et al.*, 2004).

Speeding-up the construction process allowed us to generate easily large numbers of images, which was particularly important for our adaptive local-mean control method.

4.4 Control of the local-mean histogram

4.4.1 Local-mean histogram

Our objective is to control the *local-mean* histogram of the simulated image while minimizing the continuity errors resulting from the patchwork simulation. We may want to either simulate images that reproduce the reference image statistical properties, or else simulate images that differ statistically from the reference image. If we use the transition probabilities (4.6) to perform an equiprobable nearest neighbor (ENN) simulation, then the PSM minimizes continuity errors without any control of the local-mean histogram. Our objective is to define the transition probabilities so that the local-mean histogram is controlled.

The mean value (i.e. the arithmetic average) of a state $\boldsymbol{x}$ will be denoted by $\phi(\boldsymbol{x})$. We also define the random block-state $\boldsymbol{S} \in \mathcal{S}_s$ as a block-state chosen randomly in a simulated image, assuming that all blocks are equiprobable. We choose to control the histogram of the *local-mean* $\phi(\boldsymbol{v}(\boldsymbol{S}))$ of the quarter-state $\boldsymbol{v}(\boldsymbol{S})$.

The local-mean *histogram bins* are given by $[\varphi_{i-1}, \varphi_i[$, $i = 1, 2, \ldots, M$, where the φ_i's are the bin boundaries and M is the number of bins. For the reference and simulated images, the

local-mean *bin-probabilities* will be represented by the vectors $\boldsymbol{p}_r$ and $\boldsymbol{p}$ respectively, which are defined as

$$\begin{aligned} p_r(i) &= \mathrm{P}\left[\varphi_{i-1} \leq \phi(\boldsymbol{v}(\boldsymbol{S}_r)) < \varphi_i\right] \\ p(i) &= \mathrm{P}\left[\varphi_{i-1} \leq \phi(\boldsymbol{v}(\boldsymbol{S})) < \varphi_i\right] , \end{aligned} \tag{4.9}$$

$i = 1, 2, \ldots, M$. Our objective is to simulate images such that

$$p(i) = p_t(i), \tag{4.10}$$

$i = 1, 2, \ldots, M$, where the $p_t(i)$s are user-defined *target probabilities*. It is stressed that the $p_t(i)$s *may differ* from the reference image bin-probabilities $p_r(i)$, which makes room for the simulation of images that differ statistically from the reference image. Note that $\boldsymbol{p}_t$ must satisfy the normalization constraint $\sum_{i=1}^{M} p_t(i) = 1$.

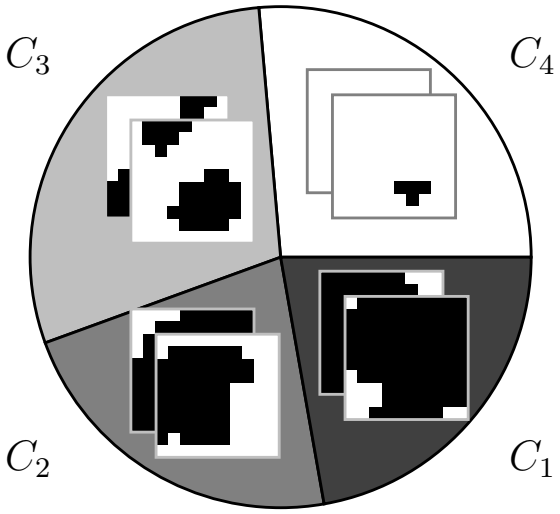


Figure 4.6 Partitioning of the reference image states $\boldsymbol{s} \in \mathcal{S}_r$ into four classes (i.e. $M = 4$) according to their local-mean. Black and white correspond to pixel values 0 and 1 respectively.

The local-mean bins allow to define a *partition* $C_1, C_2, \ldots, C_M$ of the set $\mathcal{S}_r$ (see Figure 4.6), i.e. $\bigcup_{i=1}^{M} C_i = \mathcal{S}_r$ with $C_i \cap C_j = \emptyset, \forall i \neq j$. The local-mean *classes* C_i are defined by

$$C_i = \left\{\boldsymbol{s} \in \mathcal{S}_r : \varphi_{i-1} \leq \phi(\boldsymbol{v}(\boldsymbol{s})) < \varphi_i\right\}, \tag{4.11}$$

$i = 1, 2, ..., M$, with $\varphi_M = \infty$. Thus,

$$\begin{aligned} p_r(i) &= \mathrm{P}\left[\boldsymbol{S}_r \in C_i\right] \\ p(i) &= \mathrm{P}\left[\boldsymbol{S} \in C_i\right] \end{aligned} \tag{4.12}$$

for all i. For a given simulated image, the local-mean bin-probabilities $\boldsymbol{p}$ are estimated with the *bin-frequencies* $\boldsymbol{f}$ obtained from the grid composed of all the square blocks associated to the quarter-states $\boldsymbol{v}(\boldsymbol{S}(n))$, $n = 1, 2, ..., N$, where N denotes the total number of simulation steps. The states associated to these blocks will be denoted by $\boldsymbol{V}_n$, $n = 1, 2, ..., N$. $\boldsymbol{V}_n$ is the quarter-state found in the support of $\boldsymbol{v}(\boldsymbol{S}(n))$ after the simulation is finished. The quarter-states $\boldsymbol{V}_n$ and $\boldsymbol{v}(\boldsymbol{S}(n))$ are superposed and $\boldsymbol{V}_n = \boldsymbol{v}(\boldsymbol{S}(n))$ in the absence of assembly error. However, $\boldsymbol{V}_n$ and $\boldsymbol{v}(\boldsymbol{S}(n))$ can differ if there are assembly errors in the support of $\boldsymbol{S}(n)$. The bin-frequencies are given by $f(i) = \frac{n(i)}{N}$, $i \in \{1, \ldots, M\}$, where $n(i)$ is the number of quarter-states $\boldsymbol{V}_n$ such that $\varphi_{i-1} \leq \phi(\boldsymbol{V}_n) < \varphi_i$, $n \in \{1, \ldots, N\}$.

To determine if a simulated image histogram is statistically compatible with the target probabilities, we will use the *chi-square test*. The chi-square Press *et al.* (2007) of a simulated image is evaluated with $\chi^2 = \sum_{j=1}^{M} \frac{(n(i) - N\, p_t(i))^2}{N\, p_t(i)}$.

4.4.2 The target random vector

We want the stochastic process $\boldsymbol{S}(n)$ to be *stationary* and to have the same probability distribution as a *target random vector* $\boldsymbol{S}_t$, i.e.

$$\boldsymbol{S}(n) \stackrel{\mathrm{d}}{=} \boldsymbol{S}_t \tag{4.13}$$

for each n. We want to define $\boldsymbol{S}_t$ so that the following two conditions are satisfied:

i) $\boldsymbol{S}_t$ satisfies the constraint on the local-mean histogram;

ii) $\boldsymbol{p}_t = \boldsymbol{p}_r$ implies that $\boldsymbol{S}_t \stackrel{\mathrm{d}}{=} \boldsymbol{S}_r$.

To satisfy the first condition, we impose to $\boldsymbol{S}_t$ the constraint

$$\mathrm{P}\left[\boldsymbol{S}_t \in C_i\right] = p_t(i) \tag{4.14}$$

for each i. In general, the random vector $\boldsymbol{S}_t$ is defined by its probability mass function $\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k\right]$, which can always be written in the form

$$\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k\right] = \mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k | \boldsymbol{S}_t \in C_{i(\boldsymbol{s}_k)}\right] \mathrm{P}\left[\boldsymbol{S}_t \in C_{i(\boldsymbol{s}_k)}\right], \tag{4.15}$$

where $i(\boldsymbol{s}_k)$ denotes the mean-value class index of $\boldsymbol{s}_k$. Using the constraint (4.14), (4.15) takes the form

$$\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k\right] = \mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k | \boldsymbol{S}_t \in C_{i(\boldsymbol{s}_k)}\right] p_t(i(\boldsymbol{s}_k)). \tag{4.16}$$

The probability mass function (4.16) satisfies the constraint (4.14) no matter what choice is made for the conditional probabilities $\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k | \boldsymbol{S}_t \in C_i\right]$. It seems natural to base our choice of these conditional probabilities on the reference image, i.e. we choose

$$\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k | \boldsymbol{S}_t \in C_i\right] := \mathrm{P}\left[\boldsymbol{S}_r = \boldsymbol{s}_k | \boldsymbol{S}_r \in C_i\right]. \tag{4.17}$$

Substituting (4.17) into (4.16), we obtain successively

$$\begin{aligned} \mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k\right] &= \mathrm{P}\left[\boldsymbol{S}_r = \boldsymbol{s}_k | \boldsymbol{S}_r \in C_{i(\boldsymbol{s}_k)}\right] p_t\left(i(\boldsymbol{s}_k)\right) \\ &= \frac{\mathrm{P}\left[\boldsymbol{S}_r = \boldsymbol{s}_k, \boldsymbol{S}_r \in C_{i(\boldsymbol{s}_k)}\right] p_t(i(\boldsymbol{s}_k))}{\mathrm{P}\left[\boldsymbol{S}_r \in C_{i(\boldsymbol{s}_k)}\right]} \\ &= \mathrm{P}\left[\boldsymbol{S}_r = \boldsymbol{s}_k\right] \frac{p_t(i(\boldsymbol{s}_k))}{p_r(i(\boldsymbol{s}_k))}. \end{aligned} \tag{4.18}$$

If $\boldsymbol{p}_t = \boldsymbol{p}_r$, then (4.18) implies that $\boldsymbol{S}_t \overset{\mathrm{d}}{=} \boldsymbol{S}_r$, i.e. the condition ii) is satisfied. This is our motivation for the definition (4.17).

4.4.3 Transition probabilities under stationarity hypothesis

The perfect patching approximation

At the n^{th} step of a simulation, a block-state $\boldsymbol{S}(n)$ that minimizes the distance $\| \boldsymbol{u}(\boldsymbol{S}(n)) - \boldsymbol{U}(n) \|$ is patched onto $\boldsymbol{U}(n)$. In that sense, the states $\boldsymbol{S}(n)$ and $\boldsymbol{U}(n)$ always satisfy $\boldsymbol{U}(n) \approx \boldsymbol{u}(\boldsymbol{S}(n))$

for each n. In the following derivations, it will be assumed that

$$\boldsymbol{U}(n) = \boldsymbol{u}(\boldsymbol{S}(n)) \tag{4.19}$$

for each n. The identity (4.19) will be called the *perfect patching approximation*. If the reference image is large enough, then (4.19) holds for most values of n.

Derivation of the transition probabilities

We can always write

$$\begin{aligned} p_{j,k} &= \mathrm{P}\left[\boldsymbol{S}(n) = \boldsymbol{s}_k | \boldsymbol{U}(n) = \boldsymbol{u}_j\right] \\ &= \frac{\mathrm{P}\left[\boldsymbol{S}(n) = \boldsymbol{s}_k, \boldsymbol{U}(n) = \boldsymbol{u}_j\right]}{\mathrm{P}\left[\boldsymbol{U}(n) = \boldsymbol{u}_j\right]} \\ &= \frac{\mathrm{P}\left[\boldsymbol{S}(n) = \boldsymbol{s}_k, \boldsymbol{u}(\boldsymbol{S}(n)) = \boldsymbol{u}_j\right]}{\mathrm{P}\left[\boldsymbol{u}(\boldsymbol{S}(n)) = \boldsymbol{u}_j\right]}, \end{aligned} \tag{4.20}$$

where the perfect patching approximation (4.19) was used in the third line. The hypothesis $\boldsymbol{S}(n) \stackrel{\mathrm{d}}{=} \boldsymbol{S}_t$ implies that (4.20) takes the form

$$p_{j,k} = \frac{\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k, \boldsymbol{u}(\boldsymbol{S}_t) = \boldsymbol{u}_j\right]}{\mathrm{P}\left[\boldsymbol{u}(\boldsymbol{S}_t) = \boldsymbol{u}_j\right]}. \tag{4.21}$$

The event $\boldsymbol{u}(\boldsymbol{S}_t) = \boldsymbol{u}_j$ is equivalent to the event $\boldsymbol{S}_t \in A^\dagger(\boldsymbol{u}_j)$, where the set $A^\dagger(\boldsymbol{u}_j)$ is defined by

$$A^\dagger(\boldsymbol{u}_j) := \{\boldsymbol{s} \in \mathcal{S}_r : \boldsymbol{u}(\boldsymbol{s}) = \boldsymbol{u}_j\}. \tag{4.22}$$

Substituting $\boldsymbol{u}(\boldsymbol{S}_t) = \boldsymbol{u}_j$ for $\boldsymbol{S}_t \in A^\dagger(\boldsymbol{u}_j)$ in the right hand side of (4.21), we obtain

$$\begin{aligned} p_{j,k} &= \frac{\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k, \boldsymbol{S}_t \in A^\dagger(\boldsymbol{u}_j)\right]}{\mathrm{P}\left[\boldsymbol{S}_t \in A^\dagger(\boldsymbol{u}_j)\right]} \\ &= \begin{cases} \dfrac{\mathrm{P}\left[\boldsymbol{S}_t = \boldsymbol{s}_k\right]}{\mathrm{P}\left[\boldsymbol{S}_t \in A^\dagger(\boldsymbol{u}_j)\right]} & \text{if} \quad \boldsymbol{s}_k \in A^\dagger(\boldsymbol{u}_j), \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4.23}$$

Substituting $\mathrm{P}[\boldsymbol{S}_t = \boldsymbol{s}_k]$ in (4.23) for the expression (4.18) gives

$$p_{j,k} = \begin{cases} \dfrac{\mathrm{P}[\boldsymbol{S}_r = \boldsymbol{s}_k]}{\mathrm{P}[\boldsymbol{S}_t \in A^\dagger(\boldsymbol{u}_j)]} \dfrac{p_t(i(\boldsymbol{s}_k))}{p_r(i(\boldsymbol{s}_k))} & \text{if } \boldsymbol{s}_k \in A^\dagger(\boldsymbol{u}_j), \\ 0 & \text{otherwise.} \end{cases} \tag{4.24}$$

The perfect patching approximation is not always satisfied and consequently it may happen that $A^\dagger(\boldsymbol{u}_j)$ is empty. In such cases, $\boldsymbol{S}(n)$ is chosen among the nearest neighbors of $\boldsymbol{u}_j$, i.e. in $A(\boldsymbol{u}_j)$ instead of $A^\dagger(\boldsymbol{u}_j)$. The transition probabilities that we use in practice are therefore given by

$$p_{j,k} = \begin{cases} \dfrac{\mathrm{P}[\boldsymbol{S}_r = \boldsymbol{s}_k]}{\mathrm{P}[\boldsymbol{S}_t \in A(\boldsymbol{u}_j)]} \dfrac{p_t(i(\boldsymbol{s}_k))}{p_r(i(\boldsymbol{s}_k))} & \text{if } \boldsymbol{s}_k \in A(\boldsymbol{u}_j), \\ 0 & \text{otherwise.} \end{cases} \tag{4.25}$$

It can be checked that if $\boldsymbol{U}(n) \stackrel{\mathrm{d}}{=} \boldsymbol{u}(\boldsymbol{S}_t)$, then using the transition probabilities (4.25) implies that $\boldsymbol{S}(n) \stackrel{\mathrm{d}}{=} \boldsymbol{S}_t$ (assuming perfect patching). It follows that $\boldsymbol{S}(n) \stackrel{\mathrm{d}}{=} \boldsymbol{S}_t$ for each n if $\boldsymbol{U}(n) \stackrel{\mathrm{d}}{=} \boldsymbol{u}(\boldsymbol{S}_t)$ for each n, i.e. if the random process $\boldsymbol{U}(n)$ is stationary. Simulations performed using (4.25) will be referred to as simulations with *controlled histogram under stationary assumption*, or CHUSA simulations. Model (4.25) reduces to model (4.24) if the perfect patching approximation is satisfied, i.e. if $A(\boldsymbol{u}_j) = A^\dagger(\boldsymbol{u}_j)$ for each $\boldsymbol{u}_j$. The probability $\mathrm{P}[\boldsymbol{S}_t \in A(\boldsymbol{u}_j)]$ is a normalization constant given by

$$\mathrm{P}[\boldsymbol{S}_t \in A(\boldsymbol{u}_j)] = \sum_{\boldsymbol{s}_k \in A(\boldsymbol{u}_j)} \mathrm{P}[\boldsymbol{S}_r = \boldsymbol{s}_k] \frac{p_t(i(\boldsymbol{s}_k))}{p_r(i(\boldsymbol{s}_k))}. \tag{4.26}$$

Note that if $\boldsymbol{p}_t = \boldsymbol{p}_r$, then (4.25) reduces to

$$p_{j,k} = \begin{cases} \dfrac{\mathrm{P}[\boldsymbol{S}_r = \boldsymbol{s}_k]}{\mathrm{P}[\boldsymbol{S}_r \in A(\boldsymbol{u}_j)]} & \text{if} \quad \boldsymbol{s}_k \in A(\boldsymbol{u}_j), \\ 0 & \text{otherwise.} \end{cases} \tag{4.27}$$

4.4.4 Adaptive correction of the transition probabilities

Statistical drift

Assuming that $\boldsymbol{U}(1) \stackrel{\mathrm{d}}{=} \boldsymbol{u}(\boldsymbol{S}_t)$, the probability distribution of $\boldsymbol{U}(n)$ for $n \geq 1$ may drift from the probability distribution of $\boldsymbol{u}(\boldsymbol{S}_t)$ as the simulation proceeds. This drift results from the lack of a direct control on the probability distribution of $\boldsymbol{U}(n)$. We observed that simulations produced with (4.25) are significantly better (in the statistical sense) than ENN simulations. The statistical drift of the random vectors $\boldsymbol{U}(n)$ can result in simulated images that do not respect the objective $\boldsymbol{p} = \boldsymbol{p}_t$ with sufficient accuracy. To better control the simulated images histogram, it becomes necessary to introduce correction factors in the transition probabilities (4.25). In this section, we show how these factors can be defined and adjusted adaptively to favor states that belong to the underpopulated classes C_i, and to penalize the states that belong to the overpopulated classes C_i.

A dual weighting strategy to control the statistical drift

For a given image size, low entropy images contain a smaller number of states than high entropy images. In a low entropy image, which contains relatively few states, a state $\boldsymbol{u}_j$ is typically observed many times and consequently $A(\boldsymbol{u}_j)$ typically contains several states. In a high entropy image, which contains a much larger number of states, a given state $\boldsymbol{u}_j$ typically occurs only once in the image and consequently $A(\boldsymbol{u}_j)$ typically contains only one state.

To correct for the statistical drift of the random vectors $\boldsymbol{U}(n)$, we propose to modify the transition probabilities (4.25) to modulate the probabilities $\mathrm{P}\left[\boldsymbol{S}(n) \in C_k\right]$, $k = 1, \ldots, M$.

For low entropy reference images, $A(\boldsymbol{U}(n))$ typically contains several states that belong to distinct classes C_i. In this case, the probabilities $\mathrm{P}\left[\boldsymbol{S}(n) \in C_i\right]$ can be modulated by modifying the $p_t(i)$s in (4.25). We perform this modulation by substituting $p_t(i)$ for $p'_t(i)$, which is given by

$$p'_t(i) = \frac{\omega(i)\, p_t(i)}{\sum_{k=1}^{M} \omega(k)\, p_t(k)}, \tag{4.28}$$

where the $\omega(i)$s are strictly positive weights. Increasing the value of $\omega(i)$ increases the probability $\mathrm{P}\left[\boldsymbol{S}(n) \in C_i\right]$.

For high entropy reference images, the set $A(\boldsymbol{U}(n))$ typically contains a single state. In

this case, the modulation (4.28), which exploits a random choice among several classes, does not have any effect. To modulate the transition probabilities in this case, we propose to substitute $A(\boldsymbol{u}_j)$ for the set $A'(\boldsymbol{u}_j)$ defined as

$$A'(\boldsymbol{u}_j) := \underset{\boldsymbol{s}_k \in \mathcal{S}_r}{\operatorname{argmin}} \frac{d\left(\boldsymbol{u}(\boldsymbol{s}_k), \boldsymbol{u}_j\right)}{\omega(i(\boldsymbol{s}_k))}. \tag{4.29}$$

Increasing $\omega(i)$ brings the block-states $\boldsymbol{s}_k \in C_i$ closer to $\boldsymbol{u}_j$. For high entropy reference images, $A'(\boldsymbol{U}(n))$ also contains a single state typically, like $A(\boldsymbol{U}(n))$, but this state is more likely to belong to classes with larger weights $\omega(i)$.

Since both $p'_t(i)$s and $A'(\boldsymbol{u})$ are invariant under the transformation $\boldsymbol{\omega} \to a\,\boldsymbol{\omega}$, where $a > 0$ is a constant, we impose to the weight vector $\boldsymbol{\omega}$ the constraint $\sum_{i=1}^{M} \omega(i) = 1$.

The modifications (4.28) and (4.29), which allow to modulate the probabilities $\mathrm{P}\left[\boldsymbol{S}(n) \in C_i\right]$ for both high and low entropy reference images, are complementary control mechanisms. Substituting $p'_t(i)$ for $p_t(i)$ and $A'(\boldsymbol{u}_j)$ for $A(\boldsymbol{u}_j)$ in (4.25), we obtain the following corrected transition probabilities

$$p_{j,k} = \begin{cases} \dfrac{\mathrm{P}\left[\boldsymbol{S}_r = \boldsymbol{s}_k\right]}{\alpha} \dfrac{p'_t(i(\boldsymbol{s}_k))}{p_r(i(\boldsymbol{s}_k))} & \text{if} \quad \boldsymbol{s}_k \in A'(\boldsymbol{u}_j) \\ 0 & \text{otherwise,} \end{cases} \tag{4.30}$$

where α is a normalisation constant. Simulations performed using (4.30), for which the weight vector $\boldsymbol{\omega}$ is adjusted adaptively to satisfy the objective $\boldsymbol{p} = \boldsymbol{p}_t$, will be referred to as *adaptive-CHUSA* simulations. We emphasize that low and high entropy images are extreme cases for which the relevance of the two control mechanisms (4.28) and (4.29) is easily understandable. Actually, most images are neither of low or high entropy. Model (4.30), which combines the two modulation mechanisms (4.28) and (4.29), allows our method to handle such intermediate cases.

The bin-probabilities $\boldsymbol{p}$ of images simulated using the corrected transition probabilities (4.30) depend on $\boldsymbol{\omega}$. The problem is to find an optimal weight vector $\boldsymbol{\omega}^*$ such that $\boldsymbol{p}(\boldsymbol{\omega}^*) = \boldsymbol{p}_t$. We solved this root-finding problem with an adaptation of Newton's method, as described in appendix 4.A.

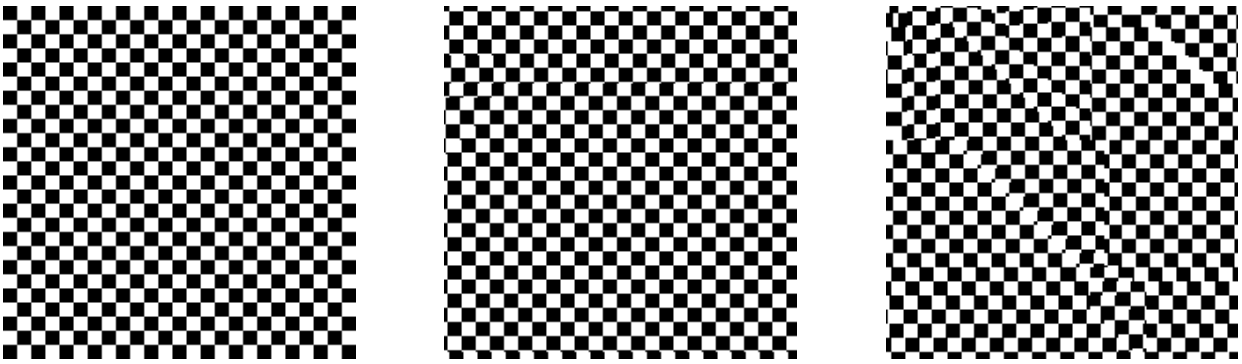
4.5 Results

We tested our simulation methods with five types of reference images. The first one is a checkerboard. The other two types are synthetic images composed of randomly located disks or channels. The last image is a scanning electron microscope photography of a polymer blend. All images are black and white, which correspond to the pixel values 0 and 1 respectively. For synthetic images, choosing black and white images composed of simple-shape objects facilitates the visual assessment of the simulation quality because the eye is effective at detecting continuity errors.

In the following simulations, the block size is $L = 16$ pixels and the substates $\boldsymbol{a}(\boldsymbol{S}(n))$ are of size $\frac{3L}{4} = 12$ pixels. The quarter-states $\boldsymbol{V}_n$ used for the control of the local-mean histogram are of size $\frac{L}{2} = 8$ pixels. All images are of size 200×200 pixels. The linear size of a simulated image before cropping is therefore $200 + L + {}^{L}\!/_{2} = 224$ pixels (see Figure 4.4h).

4.5.1 Checkerboard images

The simulated images are based on a checkerboard reference image with squares of size 8 pixels. This image allows us to assess the ability of the PSM to reproduce a periodic pattern. This example also illustrates the importance of initialization in the construction process (section 4.3). For instance, the simulated image of Figure 4.7b had initial horizontal and vertical strips well constructed, thus the PSM is able to reproduce the checkerboard nearly perfectly. In contrast, the simulated image in Figure 4.7c was intentionally initialized with a few errors along the top and left boundaries, and we observe that assembly errors propagate across the simulated image. Recall that at each step n of the initialization, an L-state $\boldsymbol{U}_k(n)$ is selected among 10 candidate L-states $\boldsymbol{U}_i(n)$ to best fit with the reference image (section 4.3). In Figure 4.7c, the initialization was done in the same way as in Figure 4.7b, except that we used only one candidate instead of ten. This example shows that it is necessary to use proper edge initialization to avoid producing biased simulations.



(a) Reference image (b) Proper edge initialisation (c) Poor edge initialisation

Figure 4.7 Images of checkerboard with squares of size 8 pixels illustrating the importance of proper edge initialization: a) reference image; b) image simulated with suitable edge initialisation; c) image simulated with poor edge initialisation.

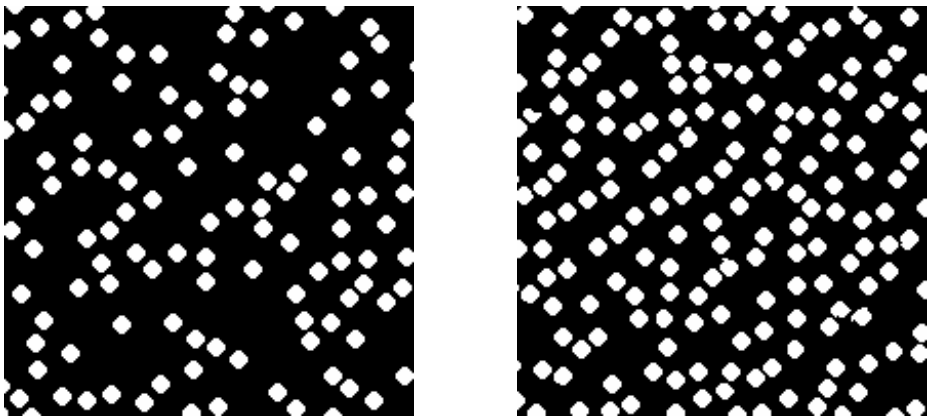
4.5.2 Images composed of disks

Simulations with $\boldsymbol{p}_t = \boldsymbol{p}_r$

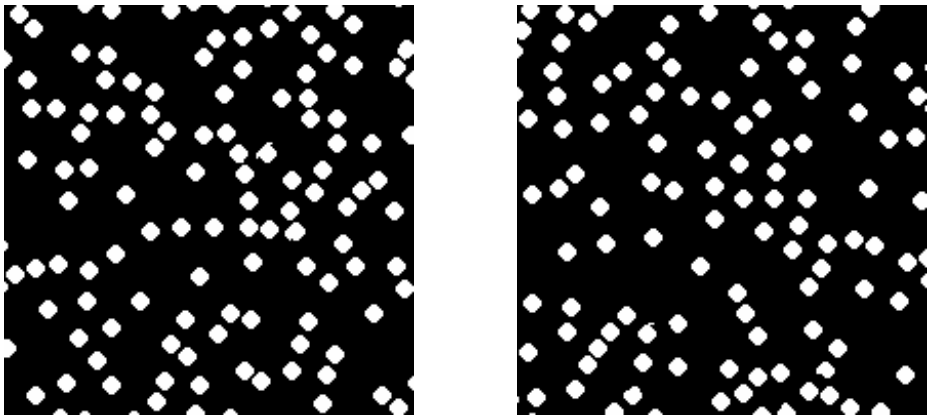
We now test the PSM on images composed of constant-diameter disjoint white disks located randomly on a uniform black background (Figure 4.8). The disks have a constant diameter of 9 pixels. To test the control of the local-mean histogram, we set the number of classes to $M = 2$ and the histogram bin boundaries to $\varphi_0 = 0$, $\varphi_1 = 1/8$, $\varphi_2 = \infty$. We choose the bin boundaries so that the bin-probabilities have similar values. We first test the case $\boldsymbol{p}_t = \boldsymbol{p}_r$, i.e. we aim at producing an image that is statistically similar to the reference image.

We produced three series of simulated images based on the reference image. The first set of $Q = 20$ images is constructed using ENN, i.e. each block-state $\boldsymbol{S}(n)$ is chosen randomly in $A(\boldsymbol{U}(n))$, assuming that all states in $A(\boldsymbol{U}(n))$ are equiprobable. The second set of 20 images are CHUSA simulations, i.e. obtained using the transition probabilities (4.25). The third set of 20 images are adaptive-CHUSA simulations, i.e. obtained using the transition probabilities (4.30) with adjusted weights. Typical simulated images are shown in Figure 4.8, where it can be observed that the PSM reproduces correctly the visual aspect of the disks, apart from a few rare imperfectly formed disks.

The local-mean histograms of the reference and simulated images are compared in Ta-



(a) Reference image (b) ENN simulation



(c) CHUSA simulation (d) Adaptive-CHUSA simulation

Figure 4.8 Images composed of disjoint disks: a) reference image; b) ENN simulation; c) CHUSA simulation; d) adaptive-CHUSA simulation.

ble 4.1. Denoting the local-mean bin frequencies of the q^{th} image by $f_q(i), i = 1, ..., M$, we computed the average bin-frequencies $f(i) := \frac{1}{Q}\sum_{q=1}^{Q} f_q(i)$ of each set of images and displayed them in the 2^{nd} and 3^{rd} columns. For instance, the reference image shows 63% of dark regions (i.e. $\mathrm{P}\left[0 \leq \phi(\boldsymbol{V}_n) < 1/8\right] = 63\%$), whereas the set of ENN simulations shows 48% of dark regions. In the fourth column of Table 4.1, the symbol χ^2 denotes the median of the set of chi-square values $\{\chi^2_q, q = 1, 2, ..., Q\}$ obtained for each image. The fifth column gives the probability P_{χ^2} that the sum of the squares of M random normal variables of unit variance and zero mean is greater than χ^2, given that the number of degrees of freedom is $M - 1$.

Table 4.1 Local-mean histogram frequencies for white disks images. The target is to reproduce the bin-probabilities of the reference image.

	$f(1)$	$f(2)$	χ^2	P_{χ^2}
Reference	63 %	37 %	NA	NA
ENN	48 %	52 %	56	~ 0 %
CHUSA	65 %	35 %	2.7	9.9 %
Ad.-CHUSA	63 %	37 %	0.82	37 %

NA stands for *not applicable*.

In Table 4.1, we observe that CHUSA simulations produce bin-frequencies that are much closer to the target probabilities than ENN simulations, and in fact nearly acceptable statistically, i.e. $P_{\chi^2} = 9.9\%$. Adaptive-CHUSA simulations produce histogram bin-frequencies that are statistically compatible with the target probabilities, i.e. $P_{\chi^2} = 37\%$.

Since our chi-square test is based on the median value χ^2 of the χ^2_q, $q \in \{1, 2, ..., Q\}$, then we expect a P_{χ^2} of approximately 50% if the simulation is statistically correct. The P_{χ^2} obtained for the adaptive-CHUSA presented in Table 4.1 and in the following tables (except for Table 4.3) are remarkably close to 50%.

We also compared the two-point statistics of the reference and simulated images via their variograms, which were estimated assuming statistical isotropy and displayed in Figure 4.9. The two CHUSA simulation methods produce a variogram that is close to the reference image variogram, and significantly better than the ENN simulation variogram.

Simulations with $\boldsymbol{p}_t \neq \boldsymbol{p}_r$

In this section, we set the target bin-probabilities to values that differ from the reference image bin-probabilities. In the first example (Figures 4.10c-d), we use target probabilities that differ by 15% from the reference image bin-probabilities, i.e. $\boldsymbol{p}_r = (55\%, 45\%)$ and $\boldsymbol{p}_t = (70\%, 30\%)$, while keeping the same bin boundaries, i.e. $\varphi_0 = 0$, $\varphi_1 = 1/8$, $\varphi_2 = \infty$.

In the second example (Figures 4.10e-f), we try a much larger change by setting the target probabilities to $p_t(1) = 90\%$ and $p_t(2) = 10\%$, that correspond to mostly black images. To reproduce the target probabilities in this case, it was necessary to use $\varphi_0 = 0$, $\varphi_1 = 1/100$, $\varphi_2 = \infty$. If φ_1 is too large, e.g. $\varphi_1 = 1/8$, then most block-states $\boldsymbol{s}$ in C_1 have a lower-right part

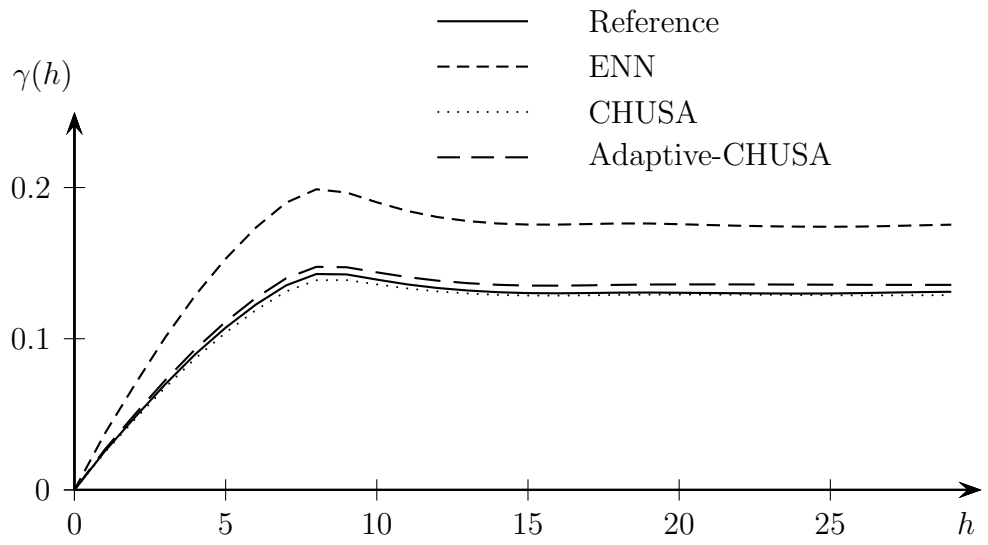


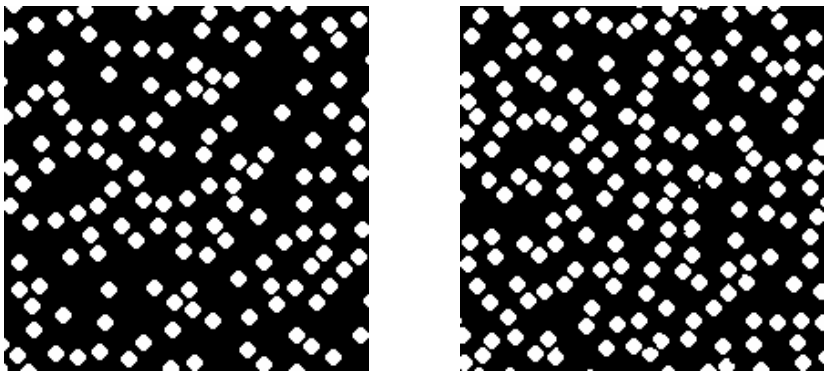
Figure 4.9 Variograms for images composed of constant diameter disks. The lag h is measured in pixels.

$\boldsymbol{v}(\boldsymbol{s})$ that contains partially constructed disks because the disk density is high in the reference image. It follows that the PSM cannot produce homogenous black regions because the PSM always tries to complete the construction of partially constructed disks. Lowering sufficiently the value of φ_1, e.g. $\varphi_1 < 1/64$, results in a class C_1 that contains block-states $\boldsymbol{s}$ for which the quarter-state $\boldsymbol{v}(\boldsymbol{s})$ is black, which allows the PSM to generate homogenous black regions if necessary.

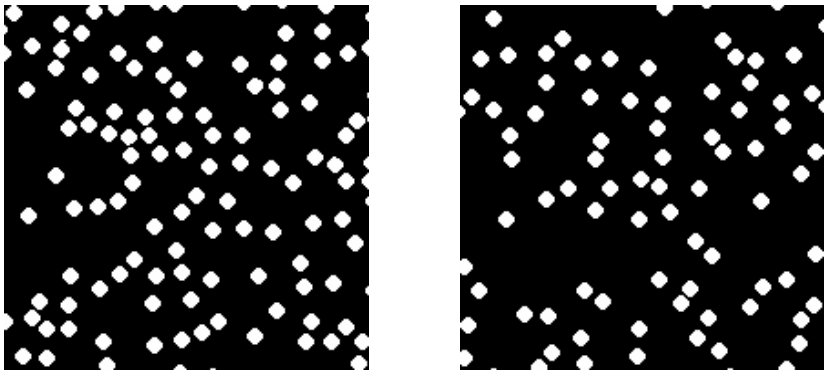
Table 4.2 Local mean histogram bin-frequencies for target probabilities with a 15% variation of the bin-probabilities to the reference.

	$f(1)$	$f(2)$	χ^2	P_{χ^2}
Reference	55 %	45 %	NA	NA
Target	70 %	30 %	NA	NA
ENN	45 %	55 %	166	~ 0 %
CHUSA	61 %	39 %	18.5	~ 0 %
Ad.-CHUSA	70 %	30 %	0.75	39 %

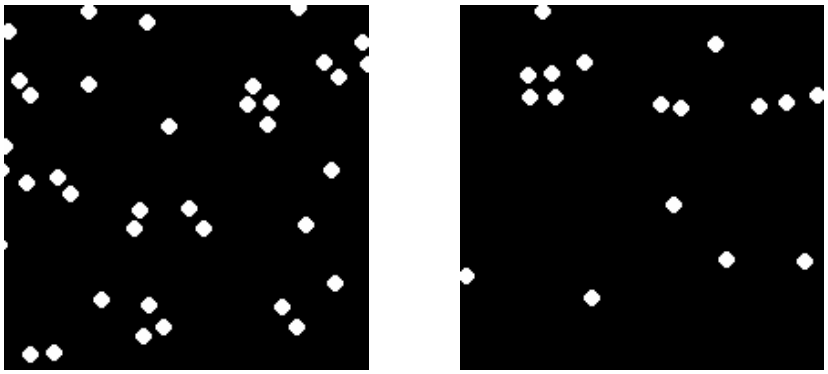
For the simulations with a 15% variation of the bin-probabilities, we observe in the Figures 4.10c-d that the disk patterns are correctly reproduced. In Table 4.2, where the histogram



(a) Reference image (b) ENN simulation



(c) CHUSA sim. No. 1 (d) Adaptive-CHUSA sim. No. 1



(e) CHUSA sim. No. 2 (f) Adaptive-CHUSA sim. No. 2

Figure 4.10 Simulations with target bin-probabilities that differ from the reference image bin-probabilities: a) reference image; b) ENN simulation; c) and e) CHUSA simulations; d) and f) adaptive-CHUSA simulations. Target probabilities are set to $(p_t(1), p_t(2)) = (70\%, 30\%)$ in c) and d), and to $(p_t(1), p_t(2)) = (90\%, 10\%)$ in e) and f).

Table 4.3 Local-mean histogram bin-frequencies for a large local-mean histogram modification.

	$f(1)$	$f(2)$	χ^2	P_{χ^2}
Reference	33 %	67 %	NA	NA
Target	90 %	10 %	NA	NA
ENN	22 %	78 %	3000	~ 0 %
CHUSA	83 %	17 %	36	~ 0 %
Ad.-CHUSA	90 %	10 %	4.7	3.0 %

bin-frequencies are compared, we observe that the CHUSA simulations are much better than ENN simulations but they do not produce statistically acceptable histograms. However, the adaptive-CHUSA simulations manage to produce images with acceptable histograms, i.e. $P_{\chi^2} = 39\%$.

For the simulations with a large variation of the bin-probabilities, we observe in the Figures 4.10e-f that the disk patterns are correctly reproduced. In Table 4.3, we observe again that the CHUSA simulations are much better than ENN simulations but they do not produce statistically acceptable histograms. In this case, even the adaptive-CHUSA does not manage to produce images with acceptable histograms, i.e. $P_{\chi^2} = 3\%$. The average frequencies $f(1)$ and $f(2)$ are correct but the chi-square test fails because the variability of the bin frequencies from one simulation to the next is too large. Minimizing continuity errors forces the adaptive-CHUSA simulations to produce images composed of disk *clusters*. The size of these clusters is not controlled and can be incompatible with the target bin probabilities.

4.5.3 Simulations with a local-mean gradient

In this section we test the ability of the PSM to produce a gradient of the local-mean histogram inside a simulated image. The simulations are based on the reference image of Figure 4.10a using $\varphi_0 = 0$, $\varphi_1 = 1/100$ and $\varphi_2 = \infty$. The goal is to produce images with a high density of disks on the left side (i.e. $\boldsymbol{p}_t = (30\%, 70\%)$) and a low density of disks on the right side (i.e. $\boldsymbol{p}_t = (90\%, 10\%)$), as shown in Figure 4.11.

For the CHUSA simulation shown in Figure 4.11a, the gradient was obtained by varying

linearly the components of the target probability vector $\boldsymbol{p}_t$ as a function of the horizontal position x between their left and right values. To obtain the adaptive-CHUSA simulation shown in Figure 4.11b, the target probability vector $\boldsymbol{p}_t$ was varied linearly in the same way as for the CHUSA simulation, but in addition the components of the weight vector $\boldsymbol{\omega}$ were also varied linearly as a function of x between their left and right values.

(a) CHUSA (b) Adaptive-CHUSA

Figure 4.11 Simulations with a gradient of the local mean: a) CHUSA simulation; b) adaptive-CHUSA simulation.

We observe in Figure 4.11 that both CHUSA and adaptive-CHUSA simulations allow to produce a gradient of the local-mean histogram. In Figure 4.12, we present the average frequency $f(1)$ as a function of the horizontal x-coordinate, where the dotted line represents a linear interpolation of $f(1)$ between its left and right values. This average frequency was computed on vertical strips for several simulated images. We observe in Figure 4.12 that both methods produce fair results.

4.5.4 Images composed of channels

In this section, we test the PSM on images that contain objects having a size much larger than the block size, e.g. channels (Figure 4.13). We chose a channel width of 6 pixels, which is much smaller than the block size $L = 16$ pixels. Unlike the reference images examined previously, these reference images are not isotropic and consequently we did not include the

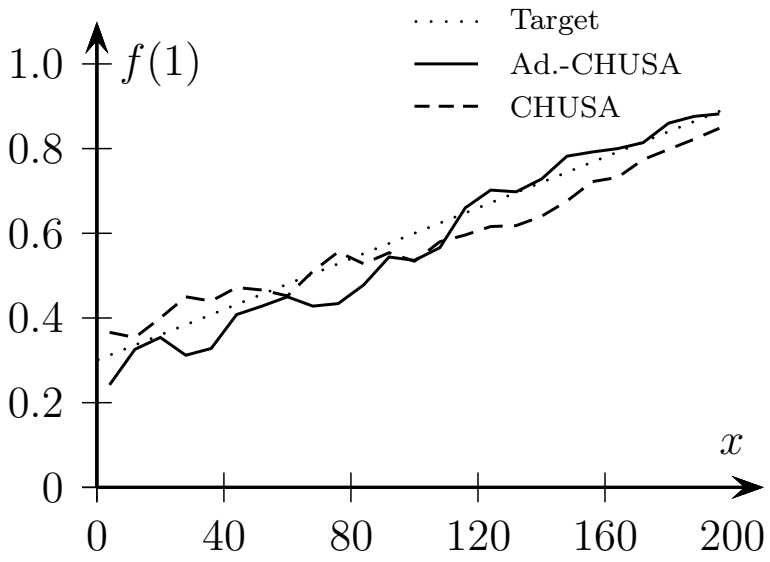


Figure 4.12 Average frequency $f(1)$ versus the horizontal x-coordinate for Figure 4.11. The coordinate x is measured in pixels.

symmetries of the reference image in the set $\mathcal{S}_r$.

The channel reference images displayed in Figure 4.13, which are strongly anisotropic, are good candidates to test the directional sensitivity of the PSM. In this perspective, we used the PSM on four reference images that differ by the general orientation of their channels: horizontal, vertical and the two diagonals (Figure 4.13, top row). The channel networks are infinite connected structures that can only stop (or start) at the image boundaries.

For the local-mean histogram control, we chose $\boldsymbol{p}_t = \boldsymbol{p}_r$, $M = 2$ and $\varphi_0 = 0$, $\varphi_1 = 1/2$, $\varphi_2 = \infty$. The histogram bin-frequencies and statistical tests are given in Table 4.4. To compare the two-point statistics, the variogram maps of the channel simulations are displayed in Figure 4.14.

In Figure 4.13, we observe in general that the visual aspect of the simulations is satisfactory and that the channels continuity is respected for most simulations. For all the simulations based on horizontal channels (Figure 4.13, first column), we observe channel networks with one or two broken branches, which can be regarded as simulation errors. These broken branches are the result of the PSM that can initiate new channels. A partially con-

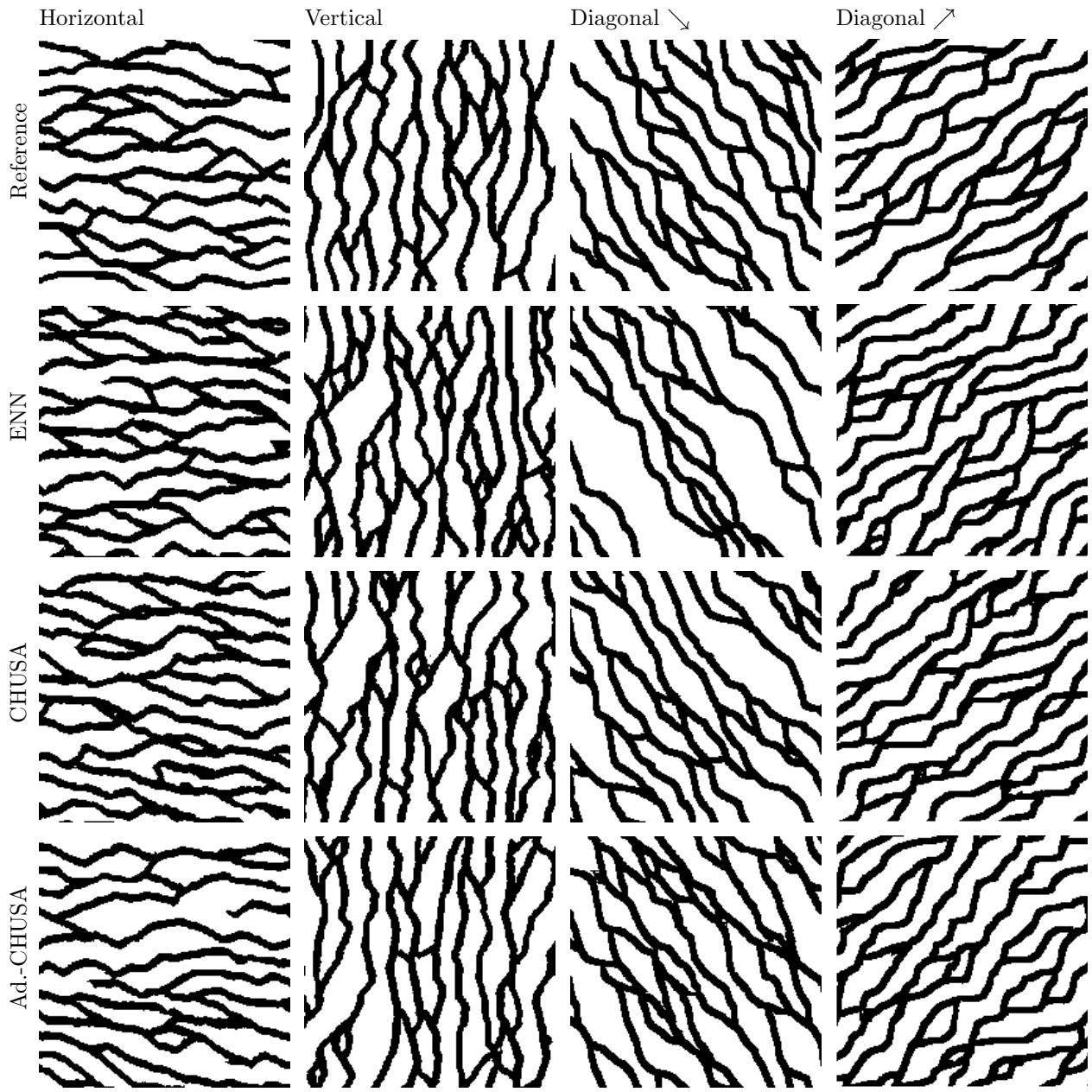


Figure 4.13 Channels with four different orientations. Each column contains the simulations for a given orientation. The first line contains the reference images with the four orientations: horizontal, vertical and the two diagonals. The other three lines contain simulated images using the simulation methods ENN, CHUSA and adaptive-CHUSA.

structed channel will be continued by the PSM in the right direction, which is the simulation direction, but will not necessarily be continued against the flow, i.e. in the left direction.

We observe in Table 4.4 that the bin-probabilities are statistically acceptable for adaptive-

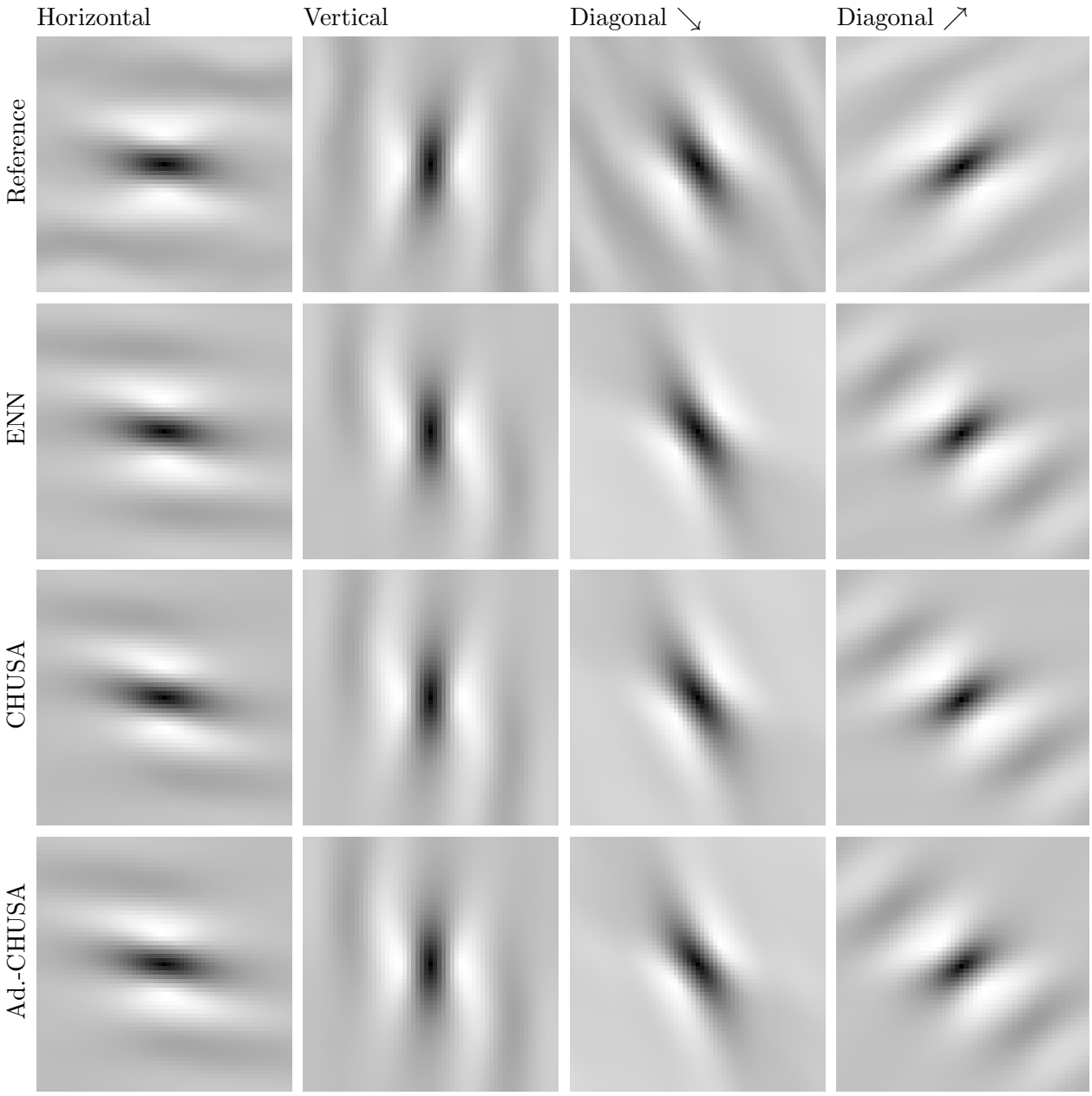


Figure 4.14 Variogram maps of the Figure 4.13.

CHUSA (i.e. $P_{\chi^2} \geq 44\%$) in all cases except for the simulations based on diagonal ↘ reference images for which $P_{\chi^2} = 4.4\%$ (third column of Figure 4.13). For these simulations, all the channel roots are set during the initialization of the construction process. If the image is initialized with a lack of channels, then a gap will propagate across the simulation. It follows that the PSM sometimes generates big blanks between channels, which results in a larger

Table 4.4 Local mean histogram bin-frequencies for channel images.

		$f(1)$	$f(2)$	χ^2	P_{χ^2}
	Reference	40 %	60 %	NA	NA
Horizontal	ENN	42 %	58 %	2.3	12.7 %
	CHUSA	43 %	57 %	5.7	1.7 %
	Ad.-CHUSA	39 %	61 %	0.59	44 %
Vertical	ENN	42 %	58 %	3.7	5.3 %
	CHUSA	41 %	59 %	1.3	25 %
	Ad.-CHUSA	39 %	61 %	0.45	50 %
Diag. ↙	ENN	32 %	68 %	32	~ 0 %
	CHUSA	33 %	67 %	24	~ 0 %
	Ad.-CHUSA	39 %	61 %	4.0	4.4 %
Diag. ↗	ENN	43 %	57 %	2.9	8.8 %
	CHUSA	42 %	58 %	2.5	11.2 %
	Ad.-CHUSA	40 %	60 %	0.41	52 %

variance of the resulting histogram.

In Figure 4.14, we observe that simulated images roughly succeed in reproducing correctly the reference images variogram maps for lags smaller than the block size L. However, the simulations variogram maps are less faithful at larger scales, especially for the two diagonal networks.

4.5.5 Simulation based on a polymer blend image

We consider a polymer blend image (Figure 4.15c) obtained by scanning electron microscope. A polymer blend is a material obtained by mixing two or more single phase polymers. The origin and preprocessing of this image are described in El Ouassini *et al.* (2006; 2008). The complex microstructure of the polymer blend is an interesting challenge for the PSM. In particular, it is well suited to test the control of the local-mean histogram at a finer level, i.e. with more than $M = 2$ classes. The polymer blend image is a high entropy image, thus CHUSA and ENN simulations give similar results. For this reason, we present only the adaptive-CHUSA simulations.

We tested the local-mean histogram control with $M = 2$ and $M = 8$. In both cases, we chose $\boldsymbol{p}_t = \boldsymbol{p}_r$, i.e. we aimed at reproducing the reference image histogram. The bin-

boundaries φ_i were chosen so that $p_t(i) \simeq 1/M$ for each i. We obtained $P_{\chi^2} = 48\%$ and 23% for $M = 2$ and $M = 8$ respectively. Both results are statistically acceptable.

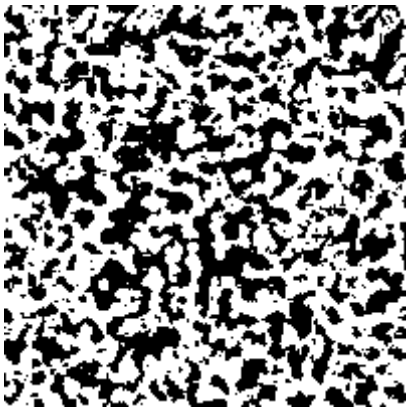
High resolution histograms (using 32 bins) of the reference image and of the simulations are shown in Figure 4.16. Comparing the Figures 4.16a and 4.16b, we observe that ENN simulations have an excess of high values, which correspond to white regions. This can be observed visually by comparing the Figures 4.15c and 4.15d. Comparing the Figures 4.16a and 4.16c, we observe that the histogram control with two classes (i.e. $M = 2$) produces a histogram that is obviously much closer to the reference image histogram than the ENN simulation histogram.

Comparing the Figures 4.16a and 4.16d, we observe that a histogram control with eight classes (i.e. $M = 8$) produces a histogram that is even better than the $M = 2$ simulation histogram. The histogram shape is correct, the histogram peak is at the right location (about 0.6) and the step located around 0.35 is well reproduced. These results show that increasing the value of M leads to a better histogram control. Comparing the simulations 4.15a and 4.15b ($M = 2$) with the simulations 4.15e and 4.15f ($M = 8$), we also observe that the latter are more faithful to the reference image 4.15c. In particular, we observe that black clusters are a bit too large for the $M = 2$ simulations (Figures 4.15a and 4.15b). We conclude that a histogram control with a higher resolution (i.e. a larger M) can improve both the histogram and the patterns geometry.

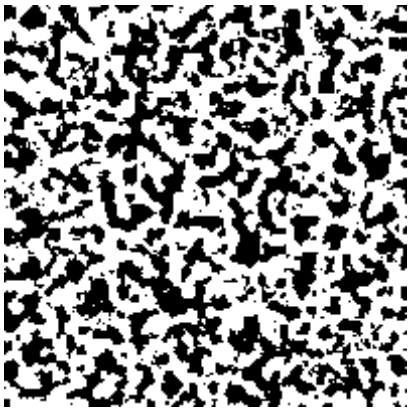
The variograms of each simulation are presented in Figure 4.17. Their comparison is visually difficult because they are quite similar, so we also plotted in Figure 4.18 the relative differences $\epsilon(h) := 100(\gamma_r(h) - \gamma_s(h))/\gamma_r(h)$, where $\gamma_r(h)$ and $\gamma_s(h)$ denote the variograms of the reference and simulated image respectively. We observe that all simulations have a variogram that differs by less that 8% from the reference. We also observe that the adaptive-CHUSA simulation with $M = 8$ produces the smallest relative differences at small scales, i.e. for $h < 7$, which confirms the superiority of these simulations.

4.5.6 Conditional simulations

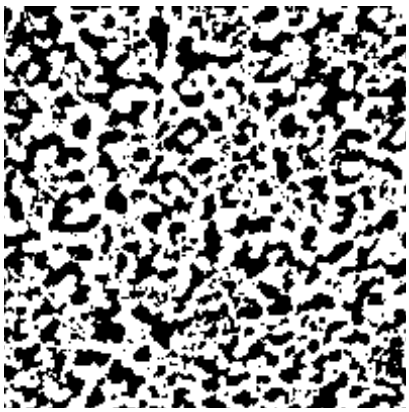
To test our simulation method with hard data, we chose the image of channels with orientation in the diagonal $\nearrow$ as the reference image (upper-right corner in Figure 4.13), and



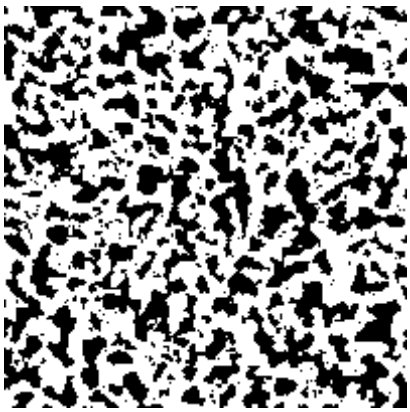
(a) Ad.CHUSA, $M = 2$



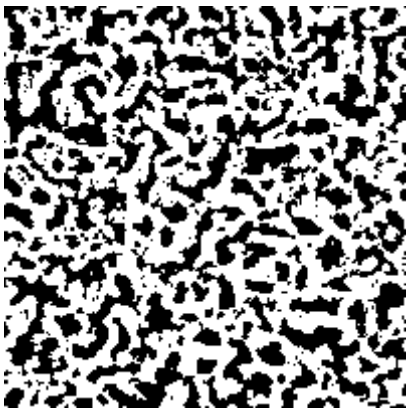
(b) Ad.CHUSA, $M = 2$



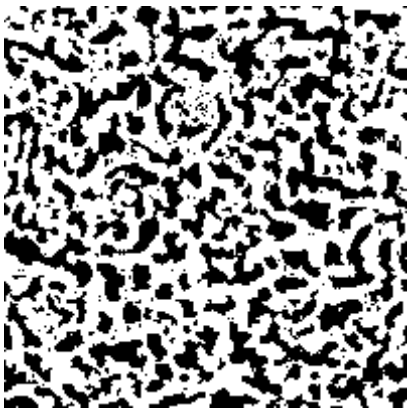
(c) Reference



(d) ENN



(e) Ad.CHUSA, $M = 8$



(f) Ad.CHUSA, $M = 8$

Figure 4.15 Polymer blend image: a) and b) adaptive-CHUSA simulations with $M = 2$; c) reference image; d) ENN simulation; e) and f) adaptive-CHUSA simulations with $M = 8$.

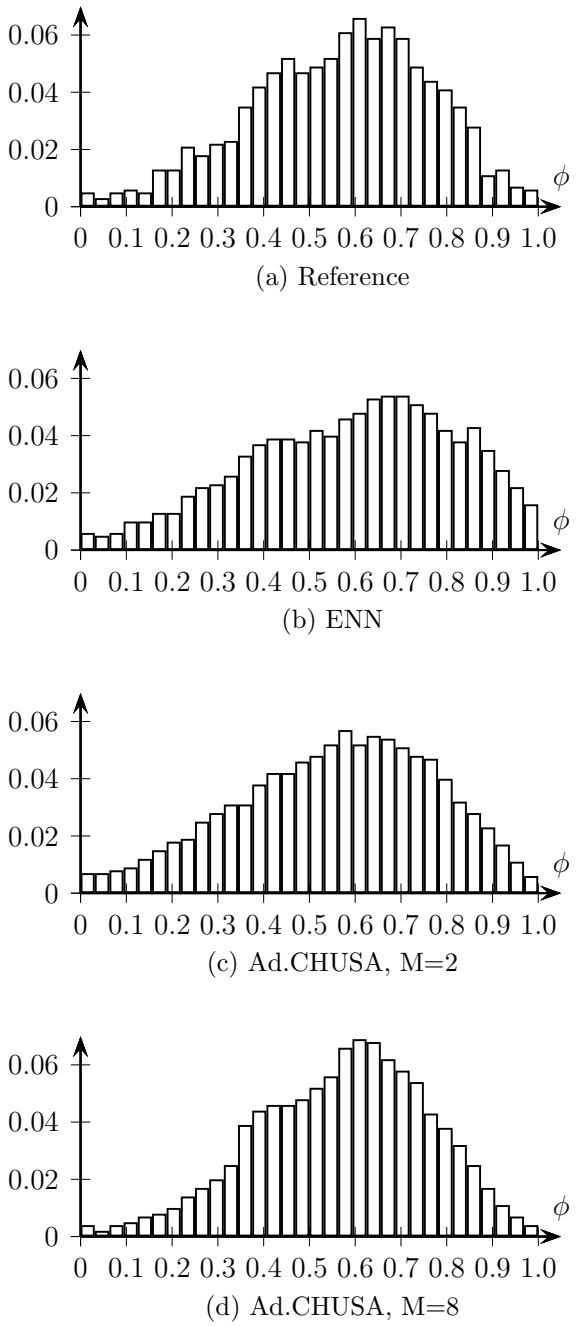


Figure 4.16 Histograms of the local-mean ϕ for polymer blend images: a) reference histogram; b) ENN simulation histogram; c) histogram of adaptive-CHUSA simulation using $M = 2$; d) histogram of adaptive-CHUSA simulation using $M = 8$.

we used adaptive-CHUSA for histogram control. We set $\boldsymbol{p}_t = \boldsymbol{p}_r$ with $M = 2$ classes and we used a block of size $L = 16$. In Figure 4.19, we display on four rows the results of four

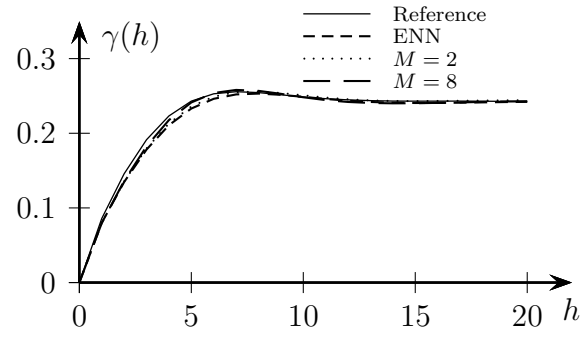


Figure 4.17 Variograms for polymer blend images. The lag h is measured in pixels.

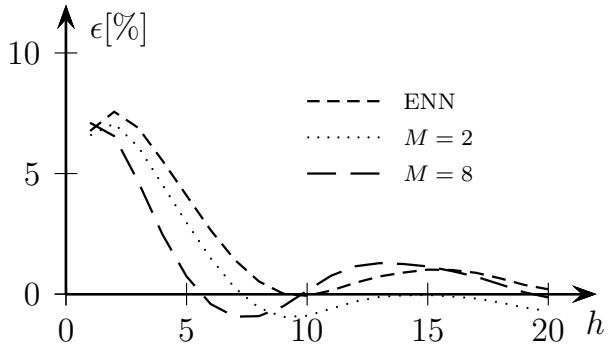


Figure 4.18 Relative variogram differences for the Figure 4.17.

different tests of our conditional simulations. The first column displays a single realization and the second column displays the E-type estimate (the average of a set of 256 simulations). The first test takes into account hard data with the short-range conditioning region, whereas all the other tests make use of the extended-range conditioning region. In all these tests, we found that the local-mean histogram is statistically acceptable, i.e. the median χ^2 is about 0.4, which corresponds to $P_{\chi^2} \approx 50\%$.

In the first test (first row in Figure 4.19), we introduced 50 hard data and we used the short-range hard data conditioning. Each hard datum is a single pixel chosen randomly and equiprobably in the reference image. In the second test (second row of Figure 4.19), we used the same 50 hard data but we used the extended-range hard data conditioning. In these two tests, we found that all hard data are respected.

On the E-type estimate of the first two tests (two top rows in Figure 4.19), we observe that the effect of a black hard datum propagates mostly in the upper-right and lower-left directions, producing an apparent shadow effect. In that sense, the effect of a hard datum on the image is anisotropic. This reflects the anisotropy of the reference image itself, in which a black pixel is most likely to have black neighbors in the upper-right and lower-left directions, given the general orientation of the channels. In contrast, the effect of a white hard datum is clearly more isotropic, which reflects the larger size of the white regions in the reference image.

On the E-type estimate of the first test (top right in Figure 4.19), we observe that each black hard datum is followed by a pale zone in the lower-right direction only. This reveals a bias in the simulation because a black channel should be surrounded by white regions on both sides, i.e. upper-left and lower-right. This bias is caused by the directionality of the unilateral simulation, i.e. left to right and top to bottom, and by the relatively short anticipation distance allowed by the short-range hard-data conditioning. On the E-type estimate of the second test (second row in Figure 4.19), we observe pale regions on both sides of each black hard datum. We conclude that the larger anticipation distance allowed by the extended-range hard-data conditioning results is a lower directional bias of the simulations. We also observe that the effect of hard-data propagates at greater distances than for the first test.

In the third test (third row of Figure 4.19), the hard data takes the form of a vertical line of white pixels at the center of the image. In this test, we found that simulations could always respect the hard data. This test reveals clearly the bias introduced by the simulation directionality. The simulation (left, third row of Figure 4.19) shows that channels are left unconnected on the left side of the white line, whereas a vertical channel tends to form in most simulations on the right side of the white line. This bias is confirmed by the E-type estimate (right, third row of Figure 4.19), where we observe a strong anisotropy of the simulations in the form of a clear difference between the left and right sides of the vertical white line.

In the fourth test (fourth row in Figure 4.19), the hard data takes the form of a solid black square of size 24×24 pixels at the center of the image. This test illustrates the difficulties encountered by our approach if one tries to fit a hard data pattern that is incompatible

with the reference image (i.e. not found in the reference image). Firstly, we found that the hard data is not fully respected by the simulation (left, fourth row of Figure 4.19). Indeed, we observe that a few white pixels remain visible inside the central square, which is unavoidable because completely black blocks of size $L = 16$ pixels do not exist in the reference image. These errors also propagate around the conditioning square block, where a few pattern continuity errors can be observed. Secondly, we found that simulations take a significantly longer time to complete (e.g. 138.9 s instead of 0.4 s for a simulation of the second row in Figure 4.19). This is explained by the fact that the algorithm, when confronted with hard data that does not exist in the reference image, scans the entire search tree in its attempt to find an image piece that fits the hard data.

Finally, we observe that a grid pattern is visible in all E-type estimates, which reveals the existence of a small bias caused by the grid used in our method. To produce negligible pattern-continuity errors at the junction between the partly simulated image and the patch-state, we observed that our algorithm has a tendency to keep channels away from this junction. It follows that junction lines are more likely to be void of channels, i.e. to be white.

4.6 Concluding remarks

We proposed an improved version of the patchwork simulation method (PSM) introduced by El Ouassini, Saucier, Marcotte and Favis (2006; 2008). This new PSM was improved in four ways.

The first improvement is that our new PSM enforces more effectively the continuity of patterns in the image. This improvement comes from a slightly different patching strategy that involves a modified weight distribution for the metric and the patching of only part of the best-fit state.

The second improvement is that our new PSM allows to control the local-mean histogram via suitably chosen transition probabilities. Our method is flexible in the sense that the target local-mean histogram can be chosen so that it differs from the reference image local-mean histogram. A major part of this work is the derivation of these transition probabilities, which were obtained in two steps. Firstly, we derived the analytical form of the transition probabilities that produce the target local-mean histogram, assuming that the stochastic

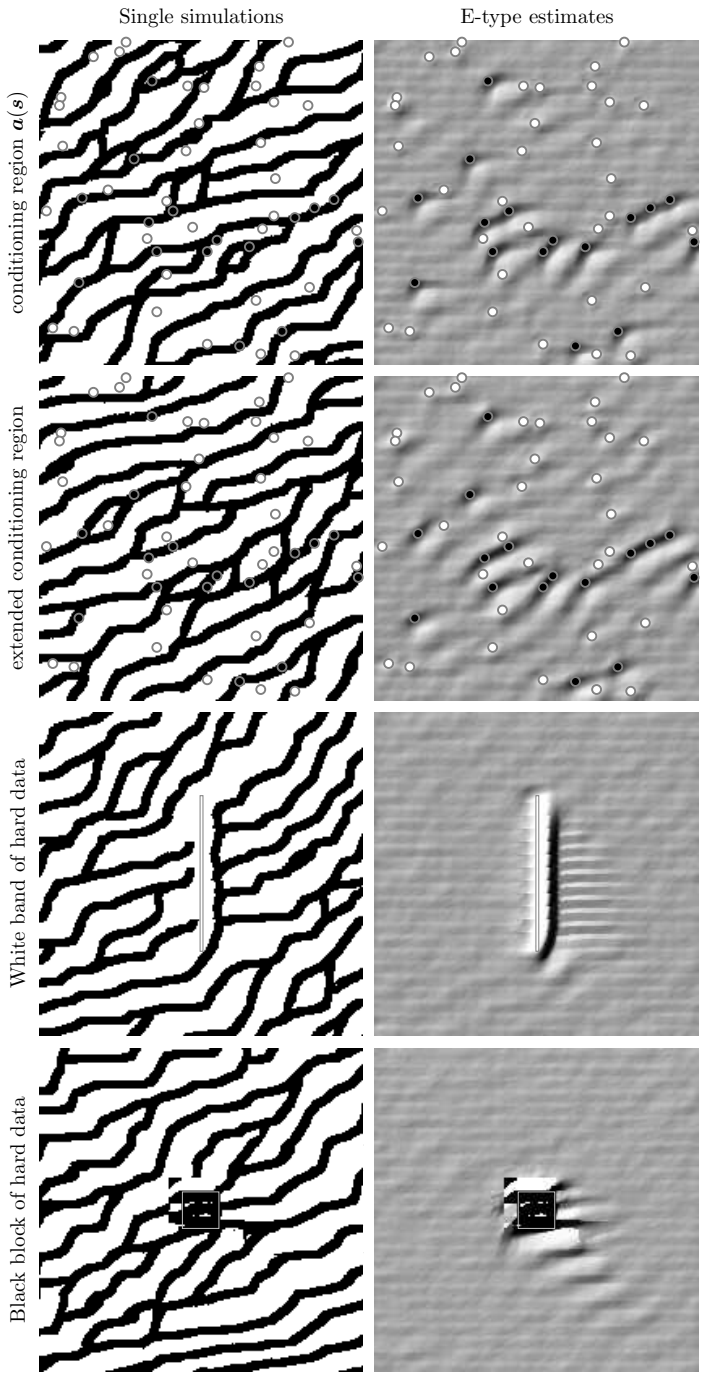


Figure 4.19 Conditional simulations for images composed of channels. In the first two rows, each hard datum (a single pixel) is encircled by a gray annulus. In the third row, the hard data take the form of a vertical line of width one pixel at the center of the image, delimited by a fine gray line. In the fourth row, the hard data form a square block at the center of the image, delimited by a fine white line.

process is stationary (CHUSA transition probabilities (4.25)). Secondly, we corrected the statistical drift of the stochastic process by introducing data-adaptive corrections to the CHUSA transition probabilities, which led to the adaptive-CHUSA transition probabilities (4.30).

The third improvement is that our new PSM makes use of a tree structure to store the states representing image pieces. This structure, which was found to be crucial to speed-up calculations, is a notable improvement on the algorithm proposed by El Ouassini *et al.* (2006; 2008). Using this tree structure allowed us to produce easily series of images for statistical tests. It also facilitated the search for the correction weights in the adaptive-CHUSA simulation method.

The fourth improvement is that our new PSM allows to make simulations that are conditional to hard data. This conditioning is achieved by restricting the choice of the patch-state to the states that honor the local hard data. The hard-data conditioning does not interfere with our control of the local-mean histogram because the adjustment of transition probabilities takes into account their presence.

It was shown that our new PSM typically succeeds in reproducing the small scale patterns found in the reference image as long as the patching square size is larger or equal to the pattern diameter.

It was shown via statistical tests that the resulting simulations actually satisfy the target local-mean histogram. It was also shown that our new PSM can be used to simulate images having non-homogeneous statistics, i.e. images for which the local-mean histogram varies as a function of the spatial position.

It was shown that the unilateral path used in our method introduces two types of bias. The first one is a directionality bias. The simulation direction creates an anisotropy that can be most easily observed in the presence of conditioning hard data. It was found that using a larger anticipation distance for the hard-data conditioning helps reducing this anisotropy, but does not eliminate it. The second type of bias is created by the grid used in the construction. It can only be observed on E-type estimates.

It should be possible to extend our simulation approach to three-dimensions because all concepts can be generalized directly, e.g. a square block in 2D becomes a cube in 3D. However,

a 3D extension of this model might raise computational issues, e.g. memory requirements of the tree structure for states storage.

Acknowledgements: The authors thank the two anonymous reviewers for their comments, which were helpful to improve significantly the original manuscript. This work was financed by the NSERC discovery research grants of the two coauthors.

4.A Determination of optimal weights

The bin-probabilities $\boldsymbol{p}$ of images simulated using the corrected transition probabilities (4.30) depend on $\boldsymbol{\omega}$. The problem is to find a weight vector $\boldsymbol{\omega}^*$ such that $\boldsymbol{p}(\boldsymbol{\omega}^*) = \boldsymbol{p}_t$. This is a root-finding problem that we tackle with a variant of Newton's method, i.e via a linearization of the function $\boldsymbol{p}(\boldsymbol{\omega})$ around a point $\boldsymbol{\omega} = \hat{\boldsymbol{\omega}}^{(0)}$. For $\left\|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}^{(0)}\right\|$ small enough, $\boldsymbol{p}(\boldsymbol{\omega})$ can be approximated by its first order Taylor series around the point $\boldsymbol{\omega} = \hat{\boldsymbol{\omega}}^{(0)}$, i.e. $\boldsymbol{p}(\boldsymbol{\omega}) \approx \boldsymbol{p}(\hat{\boldsymbol{\omega}}^{(0)}) + B\left(\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}^{(0)}\right)$ for $\left\|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}^{(0)}\right\| < \epsilon$, where $\epsilon > 0$ and B is the $M \times M$ Jacobian matrix of $\boldsymbol{p}(\boldsymbol{\omega})$ evaluated at $\boldsymbol{\omega} = \hat{\boldsymbol{\omega}}^{(0)}$. Since both $\boldsymbol{p}(\hat{\boldsymbol{\omega}}^{(0)})$ and B are unknown, it can be assumed equivalently that

$$\boldsymbol{p}(\boldsymbol{\omega}) \approx \boldsymbol{a} + B\,\boldsymbol{\omega} \quad \text{for } \left\|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}^{(0)}\right\| < \epsilon, \tag{4.31}$$

where $\boldsymbol{a} \in \mathbb{R}^M$ and B are unknown parameters. Newton's method consists in using the linear extrapolation (4.31) to estimate the root of
$\boldsymbol{p}(\boldsymbol{\omega}) - \boldsymbol{p}_t = \boldsymbol{0}$. More precisely, if $\left\|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}^{(0)}\right\| < \epsilon$ and if (4.31) holds, then $\boldsymbol{p}(\boldsymbol{\omega}) = \boldsymbol{p}_t \Rightarrow \boldsymbol{a} + B\,\boldsymbol{\omega} \approx \boldsymbol{p}_t \Rightarrow \boldsymbol{\omega} \approx B^{-1}(\boldsymbol{p}_t - \boldsymbol{a}) := \hat{\boldsymbol{\omega}}^{(1)}$, as long as B^{-1} exists. If $\hat{\boldsymbol{\omega}}^{(0)}$ is close enough to $\boldsymbol{\omega}^*$, then $\hat{\boldsymbol{\omega}}^{(1)}$ is a better estimate of $\boldsymbol{\omega}^*$ than $\hat{\boldsymbol{\omega}}^{(0)}$. This linearisation and extrapolation procedure can be repeated around the point $\boldsymbol{\omega} = \hat{\boldsymbol{\omega}}^{(1)}$ to produce an even better estimate $\hat{\boldsymbol{\omega}}^{(2)}$ of $\boldsymbol{\omega}^*$.

In the context of our problem, the difficulty is that the function $\boldsymbol{p}(\boldsymbol{\omega})$ is not known. At each step of the method, we must *estimate* the parameters $\boldsymbol{a}$ and B by performing a multidimensional linear regression in the neighborhood of the best estimate of $\boldsymbol{\omega}^*$. At the n^{th} step, we choose a neighborhood $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$ that has the shape of a rectangular parallepiped centered on $\hat{\boldsymbol{\omega}}^{(n-1)}$, and for which the widths in each direction are specified by

the array $\Delta\boldsymbol{\omega} := (\delta\omega, ..., \delta\omega)^T$, where $\delta\omega = \frac{\alpha}{M}$ and $\alpha \in]0, 1[$ is an adjustable parameter. The width $\delta\omega$ must small enough for $\boldsymbol{p}(\boldsymbol{\omega})$ to be approximately linear within $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$.

The estimation of $(\boldsymbol{a}, B)$ at the n^{th} step is based on $Q(n)$ simulated images generated using distinct weight vectors $\boldsymbol{\omega}_q, q = 1, ..., Q(n)$, which all belong to the box $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$. For each image we compute the *bin-frequencies* $\boldsymbol{f}_q(\boldsymbol{\omega}_q)$, which are estimates of the bin probabilities $\boldsymbol{p}(\boldsymbol{\omega}_q)$. Since $\boldsymbol{f}_q(\boldsymbol{\omega}_q)$ and $\boldsymbol{p}(\boldsymbol{\omega}_q)$ differ by a random fluctuation, it follows from (4.31) that $\boldsymbol{f}_q(\boldsymbol{\omega}_q)$ is related to $\boldsymbol{\omega}_q$ by the model

$$\boldsymbol{f}_q(\boldsymbol{\omega}_q) = \boldsymbol{a} + B\,\boldsymbol{\omega}_q + \boldsymbol{W}_q, \quad q = 1, ..., Q(n), \tag{4.32}$$

where $\boldsymbol{W}_q \in \mathbb{R}^M$ is a random noise that includes both the random fluctuation and the linearization error. Using model (4.32), the least-squares estimate $(\hat{\boldsymbol{a}}, \hat{B})$ of $(\boldsymbol{a}, B)$ is

$$(\hat{\boldsymbol{a}}, \hat{B}) = \underset{\boldsymbol{a}, B}{\operatorname{argmin}} \quad \frac{1}{2} \sum_{q=1}^{Q(n)} \left\| \boldsymbol{f}_q(\boldsymbol{\omega}_q) - \boldsymbol{a} - B\,\boldsymbol{\omega}_q \right\|^2. \tag{4.33}$$

The multidimensional linear regression used to obtain $(\hat{\boldsymbol{a}}, \hat{B})$ from (4.33) is discussed in the appendix 4.B. Note that the condition $Q(n) \geq M + 1$ must be satisfied for each n for the problem (4.33) to be well posed. To reduce the variance of $\hat{\boldsymbol{a}}$ and $\hat{B}$, we use $Q(n) \geq Q_{\min} > M + 1$, where the value $Q_{\min} := M + 20$ was used in our simulations. The resulting estimate $\hat{\boldsymbol{\omega}}^{(n)}$ of $\boldsymbol{\omega}^*$ is

$$\hat{\boldsymbol{\omega}}^{(n)} = \hat{B}^{-1} \left(\boldsymbol{p}_t - \hat{\boldsymbol{a}} \right). \tag{4.34}$$

At each step, the weight vector $\boldsymbol{\omega}_q$ must be suitably chosen. The iteration is initiated with uniform weights, i.e. with $\hat{\boldsymbol{\omega}}^{(0)} = (1/M, ..., 1/M)^T$. Note that the choice $\boldsymbol{\omega} = \hat{\boldsymbol{\omega}}^{(0)}$ usually gives a fair approximation of $\boldsymbol{\omega}^*$ in the case where $\boldsymbol{p}_t = \boldsymbol{p}_r$. In the first iteration, i.e. for $n = 1$, we generate randomly $Q(1) = Q_{\min}$ values of $\boldsymbol{\omega}$ that lie in the box $\Omega(\hat{\boldsymbol{\omega}}^{(0)}, \Delta\boldsymbol{\omega})$. More precisely, the $\boldsymbol{\omega}_q$s are generated with

$$\boldsymbol{\omega}_q = \frac{\hat{\boldsymbol{\omega}}^{(n-1)} + \delta\omega\,\boldsymbol{\epsilon}_q^{(n)}}{\sum_{i=1}^{M} \hat{\omega}^{(n-1)}(i) + \delta\omega\;\epsilon_q^{(n)}(i)} \tag{4.35}$$

$q = 1, ..., Q(1)$, where the $\epsilon_q^{(n)}(i)$s are mutually independent random variables uniformly dis-

tributed in $[-1/2, 1/2]$. The denominator in (4.35) insures that the normalization constraint on the weights is satisfied. The estimation is performed from the values of $\boldsymbol{\omega}$ contained in the list $\mathbb{L}_1 := \{\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_{Q(1)}\}$, which leads to the first estimate $\hat{\boldsymbol{\omega}}^{(1)}$.

In the next iterations, i.e. for $n \geq 2$, the updated box $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$ is centered on the latest estimate $\hat{\boldsymbol{\omega}}^{(n-1)}$ and contains a subset of the $\boldsymbol{\omega}_q$s contained in $\mathbb{L}_{n-1}$. Indeed, the box displacement may cause the exclusion of some $\boldsymbol{\omega}_q$s. If the number k of $\boldsymbol{\omega}_q$s in $\mathbb{L}_{n-1} \cap \Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$ is less than $Q_{\min}$, then we generate randomly a complementary list $\mathbb{L}_n^*$ of at least $Q_{\min} - k$ new values of $\boldsymbol{\omega}$ contained in $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$. If $k \geq Q_{\min}$, then we generate only one additional value of $\boldsymbol{\omega}$ contained in $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$. In both cases, the new values of $\boldsymbol{\omega}$ are generated via (4.35). The new estimate $\hat{\boldsymbol{\omega}}^{(n)}$ is computed from the values of $\boldsymbol{\omega}$ contained in the updated list $\mathbb{L}_n$ defined by

$$\mathbb{L}_n := \{\boldsymbol{\omega}_q \in \mathbb{L}_{n-1} \cap \Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})\} \cup \mathbb{L}_n^*, \tag{4.36}$$

which always contains at least $Q_{\min}$ elements. Note that we do not include in $\mathbb{L}_n$ the estimates $\hat{\boldsymbol{\omega}}^{(1)}, \hat{\boldsymbol{\omega}}^{(2)}, ..., \hat{\boldsymbol{\omega}}^{(n-1)}$, which results in a homogeneous distribution of the $\boldsymbol{\omega}$s in $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$.

As n increases, the box movement slows down and comes to a near-stop around the optimal solution. For n large enough, the box position is nearly constant and the number of $\boldsymbol{\omega}_q$s in $\Omega(\hat{\boldsymbol{\omega}}^{(n-1)}, \Delta\boldsymbol{\omega})$ increases. We stop iterating when $|\mathbb{L}_n| = Q(n) = Q_{\max}$, which occurs for $n = n_{\max}$, and we keep $\hat{\boldsymbol{\omega}}^{(n_{\max})}$ as the best estimate of $\boldsymbol{\omega}^*$. The value of $Q_{\max}$ depends on the accuracy needed for the estimate $\boldsymbol{\omega}^*$.

Note that the box size $\delta\omega = \frac{\alpha}{M}$ must be small enough for the linear model (4.32) to be valid within the box. To test this condition, we repeat the estimation process with a series of decreasing values of α, e.g. $\{1/2, 1/4, 1/8, ..., 1/2^n, ...\}$, and we stop when the estimate of $\boldsymbol{\omega}^*$ no longer changes significantly with α.

4.B Estimation of a and B

To find the least-squares estimates $(\hat{\boldsymbol{a}}, \hat{B})$ of the parameters $(\boldsymbol{a}, B)$, we search for the minimum of the function

$$\begin{aligned} E(\boldsymbol{a}, B) &= \tfrac{1}{2} \sum_{q=1}^{Q} \|\hat{\boldsymbol{p}}_q - \boldsymbol{a} - B\,\boldsymbol{\omega}_q\|^2 \\ &= \tfrac{1}{2} \sum_{q=1}^{Q} \sum_{i=1}^{M} \Big(\hat{p}_q(i) - a(i) - \sum_{j=1}^{M} b_{i,j}\,\omega_q(j)\Big)^2 \end{aligned} \tag{4.37}$$

with respect to $\boldsymbol{a}$ and B. In (4.37), the $b_{i,j}$s are the elements of the Jacobian matrix B. The unknown parameters are the M components of $\boldsymbol{a}$ and the M^2 matrix elements of B, for a total of $M + M^2 = M(M+1)$ unknowns. Note that the minimum value $E(\boldsymbol{a}, B) = 0$ can be obtained if

$$\hat{\boldsymbol{p}}_q - \boldsymbol{a} - B\,\boldsymbol{\omega}_q = 0,\ q \in \{1, 2, ..., Q\}. \tag{4.38}$$

The vectors in (4.38) are M-dimensional, hence (4.38) is a linear system of $Q\,M$ equations that has a unique solution if the number of equations is equal to the number of unknowns, i.e. if $Q\,M = M(M+1) \Rightarrow Q = M + 1$. Hence we expect the minimization problem (4.33) to be uniquely determined if $Q = M + 1$, in which case $E(\hat{\boldsymbol{a}}, \hat{B}) = 0$, and over-determined if $Q > M + 1$, in which case $E(\hat{\boldsymbol{a}}, \hat{B}) \geq 0$.

The gradient of E vanishes at the minimum of E. The conditions $\frac{\partial E}{\partial a_i} = 0$ and $\frac{\partial E}{\partial b_{i,j}} = 0$ lead respectively to the equations

$$\begin{aligned} &Q\,a(i) + \sum_{j=1}^{M} b_{i,j} \sum_{q=1}^{Q} \omega_q(j) = \sum_{q=1}^{Q} \hat{p}_q(i) \\ &a_i \sum_{q=1}^{Q} \omega_q(j) + \sum_{k=1}^{M} b_{i,k} \sum_{q=1}^{Q} \omega_q(k)\,\omega_q(j) = \sum_{q=1}^{Q} \hat{p}_q(i)\,\omega_q(j) \end{aligned} \tag{4.39}$$

for $i, j \in \{1, \ldots, M\}$. The equations (4.39) can be rewritten in the equivalent matrix form

$$\begin{pmatrix} Q & \sum_{q=1}^{Q} \boldsymbol{\omega}_q^T \\ \sum_{q=1}^{Q} \boldsymbol{\omega}_q & \sum_{q=1}^{Q} \boldsymbol{\omega}_q\,\boldsymbol{\omega}_q^T \end{pmatrix} \begin{pmatrix} \boldsymbol{a}^T \\ B^T \end{pmatrix} = \begin{pmatrix} \sum_{q=1}^{Q} \hat{\boldsymbol{p}}_q^T \\ \sum_{q=1}^{Q} \boldsymbol{\omega}_q\,\hat{\boldsymbol{p}}_q^T \end{pmatrix}. \tag{4.40}$$

If $Q \geq M + 1$, the linear systems of equations (4.40) can be solved numerically for $\boldsymbol{a}$ and B.

CHAPITRE 5

ARTICLE 2: CORRECTIVE PATTERN-MATCHING SIMULATION WITH CONTROLLED LOCAL-MEAN HISTOGRAM

Corentin Faucher

École Polytechnique de Montréal, Mathématiques et génie industriel, C.P. 6079, Succursale Centre-ville, Montréal (Québec), Canada, H3C-3A7.

e-mail: corentin.faucher@polymtl.ca

Antoine Saucier

École Polytechnique de Montréal, Mathématiques et génie industriel, C.P. 6079, Succursale Centre-ville, Montréal (Québec), Canada, H3C-3A7.

Tel.: (1) 514-340-4711 ext. 4516

Fax: (1) 514-340-4463

e-mail: antoine.saucier@polymtl.ca

Denis Marcotte

École Polytechnique de Montréal, Génies civil, géologique et des mines, C.P. 6079, Succursale Centre-ville, Montréal (Québec), Canada, H3C-3A7.

e-mail: denis.marcotte@polymtl.ca

Article soumis le 12 septembre 2012 chez Springer dans la revue *Mathematical Geosciences*.

Abstract

We present a new stochastic simulation method that builds two-dimensional images by assembling together square image pieces called blocks. The blocks are taken from a reference image that provides the statistical information on the simulated field. Our method, called corrective pattern-matching simulation, proceeds iteratively by making local corrections to

the simulated image. Our method favors pattern continuity in the image, respects conditional hard data and allows to control the image local-mean histogram. This histogram can be set to a user-defined target value which may differ from the reference image local-mean histogram. For several types of images, we show that our simulations respect the conditioning data, reproduce faithfully the reference image visual appearance and are statistically compatible with a target local-mean histogram. It is also shown that our method does not have the anisotropy biases that are associated with the unilateral patchwork simulation method.

Keywords simulations with patterns •reference image •histogram control •pattern similarity •multiple-point statistics •geostatistics •greedy algorithm

5.1 Introduction

Simulations taking into account multiple-point statistics are used in geostatistics to model the spatial distribution of geological formations (e.g. oil reservoirs) and to reproduce accurately their geometrical properties. Sequential simulation approches that take into account multiple-point statistics can be categorized in two classes. The first class contains methods that build the image one node at a time by means of conditional probabilities. The latter are estimated from a training image (Guardiano and Srivastava, 1993; Strebelle, 2002; 2003).

In these methods, the simulation proceeds sequentially on a random path. The second class contains methods for which the basic building block of the image is a multiple-nodes structure called a *pattern*. A pattern is patched to the image at each simulation step. For these simulations, which have been called *simulation with patterns* (Arpat and Caers, 2007; 2005; Zhang, Switzer and Journel, 2006) or *patchwork simulations* (El Ouassini, Saucier, Marcotte, Faucher and Favis, 2006; 2008; 2012) the image is constructed sequentially like a jigsaw puzzle, i.e. by assembling together image pieces. The advantage of these methods is that the patterns contained in each image piece are correct by definition because they are taken from a reference image that is assumed to be representative of the random field to simulate. This advantage also holds for the simulation if the assembly process preserves the image pieces integrity and if the image pieces are suitably assembled.

The conditional simulation with patterns (Arpat and Caers, 2005) proceeds by sticking a pattern (a $L \times L$ square-shaped image-piece) at each visited location along a random path

that is independent of the image state. The pattern is taken in a training image to fit the previously simulated data and the conditioning hard data. This method makes use of a multiple-grid structure to simulate the image from the coarsest to the finest grid. The patchwork simulation method (PSM) proceeds by assembling square-shaped image pieces called blocks in a unilateral order, i.e. left to right and top to bottom. At each simulation step, transition probabilities are used to determine which block is chosen in a reference image. Patchwork simulations have a resemblance with a Markov mesh random field (Pickard, 1980; Razlighi, Kehtarnavaz and Nosratinia, 2009) because transition probabilities are used to associate a new block to existing neighbor blocks. A recent version (Faucher, Saucier and Marcotte, 2012) of the patchwork simulation method allows to control the simulation local-mean histogram.

The ability to reproduce one-point statistics such as the mean is important in geostatistical simulations. For example, the facies proportions in an oil reservoir may be known approximately, e.g. via seismic data. In this context, simulations of this reservoir should reproduce both the facies proportions and the patterns observed in a reference image.

Stochastic sequential simulations used to model geological formations are typically constrained to respect conditional *hard data*. It was found that the sequential construction method used in the patchwork simulation method (Faucher *et al.*, 2012) is not ideally suited to tackle the constraints defined by hard data. It was also found that the directional aspect of the PSM creates an anisotropy that can be easily observed in the presence of conditioning hard data. This leads us to believe that the image regions that are subject to many constraints should be constructed first, whereas regions subject to less constraints should be constructed second. To tackle hard data more efficiently, we introduce in this paper a new simulation method called *corrective pattern-matching simulation* (CPMS). This method constructs the image in a non-unilateral order: the method starts by constructing the image in the regions that contain hard data, and then continues the image construction in the remaining regions.

The CPMS is primarily inspired from the patchwork simulation method (Faucher *et al.*, 2012). As in the PSM, the CPMS proceeds sequentially by patching blocks from the reference image to the simulated image. The CPMS also allows to control the local-mean histogram of

the simulated image. This control is made by suitably selecting the blocks that are patched to the simulated image. However, the CPMS differs from the PSM in some crucial aspects. Firstly, the CPMS is not unilateral. The location at which a block is corrected is selected randomly with a probability proportional to a local error. This local error actually takes into account the three factors that we want to control: the respect of hard data, the respect of a target local-mean histogram and the faithful reproduction of the patterns observed in the reference image. Secondly, the CPMS does not make use of any transition probabilities. Instead, the algorithm makes a suitable choice of the block that fits best locally, taking into account all constraints (i.e. hard data, local-mean target histogram and reproduction of the reference image patterns). Patching the chosen block into the simulated image achieves a local *correction* of the image, which motivates our designation of *corrective pattern-matching simulation*.

There are also similarities between the CPMS and the conditional simulation with patterns of Arpat and Caers (2007), which constructs images by patching blocks that respect both the local hard data and the available local data. In contrast with the method of Arpat and Caers (2007), the path followed by CPMS during the construction process is not random, it rather visits in priority the locations where the local error is large. Arpat and Caers used a multiple grid to reproduce patterns at multiple scales, whereas the CPMS ensures the control of the local-mean histogram at a given scale L and focuses on patterns continuity in the image.

5.2 Overview of the simulation method

5.2.1 Problem statement

The CPMS constructs a *simulated image* by assembling together square image pieces extracted from a *reference image*. These image pieces contain patterns that are assumed to be representative of the random field to simulate. We want the simulated image to honor the following three objectives:

– satisfy conditioning data;

– reproduce the reference image patterns;

– ensure that the simulation local-mean histogram matches statistically a user-given target histogram.

Let $\boldsymbol{x} = (x(1), x(2), ..., x(Q))^T$ denote an array that contains all the pixel values of the simulated image. For simplicity, all images considered in this paper are composed of only two facies, i.e. each pixel value $x(p)$ is equal to 0 (black) or 1 (white), where $p \in \{1, 2, ..., Q\}$, i.e. $\boldsymbol{x} \in X := \{0, 1\}^Q$. The objective of the CPMS is to construct an image $\boldsymbol{x}$ that is a solution of the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{x} \in X}{\text{minimize}} \quad & E_{\text{pc}}(\boldsymbol{x}) \\ \text{subject to} \quad & E_{\text{hd}}(\boldsymbol{x}) = 0, \\ & D^2(\boldsymbol{x}) \leq R^2. \end{aligned} \tag{5.1}$$

In (5.1), the *pattern-conformity error function* $E_{\text{pc}}(\boldsymbol{x}) \geq 0$ is a measure of the dissimilarity of the patterns found in the simulated image to the patterns found in the reference image. A smaller value of $E_{\text{pc}}(\boldsymbol{x})$ corresponds to an image that contains patterns that are closer to the reference image patterns. The *hard data error function* $E_{\text{hd}}(\boldsymbol{x}) \geq 0$ is proportional to the number of pixels in the simulated image that do not honor hard data, hence $E_{\text{hd}}(\boldsymbol{x}) = 0$ if and only if all hard data are respected. The quantity $D^2(\boldsymbol{x}) \geq 0$ is a statistics associated to the chi-square test that we use to quantify the respect of the target local-mean histogram (using M bins, $M \geq 2$). The condition $D^2(\boldsymbol{x}) \leq R^2$, where R^2 is a significance threshold to be defined, ensures that the simulation respects the target local-mean histogram.

The dimension of $\boldsymbol{x}$ is large, e.g. $200 \times 200 = 40000$, and the components of $\boldsymbol{x}$ take only the two integer values 0 and 1. Moreover, it will be seen in section 5.3.2 that the function $E_{\text{pc}}(\boldsymbol{x})$ is not differentiable for all $\boldsymbol{x} \in \mathbb{R}^Q$. This situation implies that obtaining exact solutions of problem (5.1) is difficult. For these reasons, we tackle this problem with a heuristic method.

5.2.2 Solution strategy

Auxiliary optimization problem

We introduce the following *auxiliary* optimization problem:

$$\underset{\boldsymbol{x}\in X}{\text{minimize}}\ E(\boldsymbol{x}), \tag{5.2}$$

where

$$E(\boldsymbol{x}) := E_{\text{pc}}(\boldsymbol{x}) + \mu_1\, E_{\text{hd}}(\boldsymbol{x}) + \mu_2\, E_{\text{stat}}(\boldsymbol{x}) \tag{5.3}$$

is defined as the *global error*, whereas μ_1 and μ_2 are positive penalty coefficients. The constraints of problem (5.1) are introduced in (5.2) in the form of the penalty functions $E_{\text{hd}}(\boldsymbol{x})$ and $E_{\text{stat}}(\boldsymbol{x})$. The *statistical error function* $E_{\text{stat}}(\boldsymbol{x}) \geq 0$, which differs from $D^2(\boldsymbol{x})$ in (5.1), is defined in such a way (section 5.3.3) that a decrease of $E_{\text{stat}}(\boldsymbol{x})$ implies a decrease of $D^2(\boldsymbol{x})$, and so that $E_{\text{stat}}(\boldsymbol{x}) \to 0 \Rightarrow D^2(\boldsymbol{x}) \to 0$. Using suitable values for μ_1 and μ_2, a vector $\boldsymbol{x}$ that produces a small value of $E(\boldsymbol{x})$ will also produce small values of the three functions $E_{\text{pc}}(\boldsymbol{x})$, $E_{\text{hd}}(\boldsymbol{x})$ and $E_{\text{stat}}(\boldsymbol{x})$, and therefore will be an approximate solution of problem (5.1).

Starting with an unstructured initial candidate solution $\boldsymbol{x}$ (we use a white noise for which each pixel takes the values 0 or 1 with equal probabilities), our approach is to improve $\boldsymbol{x}$ iteratively with respect to a measure of quality defined by the function $E(\boldsymbol{x})$. Our approach to problem (5.2) is therefore a *metaheuristic*. It will be shown that suitable choices of μ_1 and μ_2 as well as a suitable stochastic improvement scheme typically lead to satisfactory solutions of problem (5.1). This improvement scheme consists in using the three functions $E_{\text{pc}}(\boldsymbol{x})$, $E_{\text{hd}}(\boldsymbol{x})$ and $E_{\text{stat}}(\boldsymbol{x})$ to construct a suitable direction of descent at point $\boldsymbol{x}$, which typically results in a decrease of the global error $E(\boldsymbol{x})$. To define our metaheuristic, we must first define the functions $E_{\text{pc}}(\boldsymbol{x})$, $E_{\text{hd}}(\boldsymbol{x})$ and $E_{\text{stat}}(\boldsymbol{x})$.

Boxes, covering and states

The support of an image is a rectangular region. We define a *box* to be a square region positioned in the image support so that it contains only entire pixels (pixels are squares of

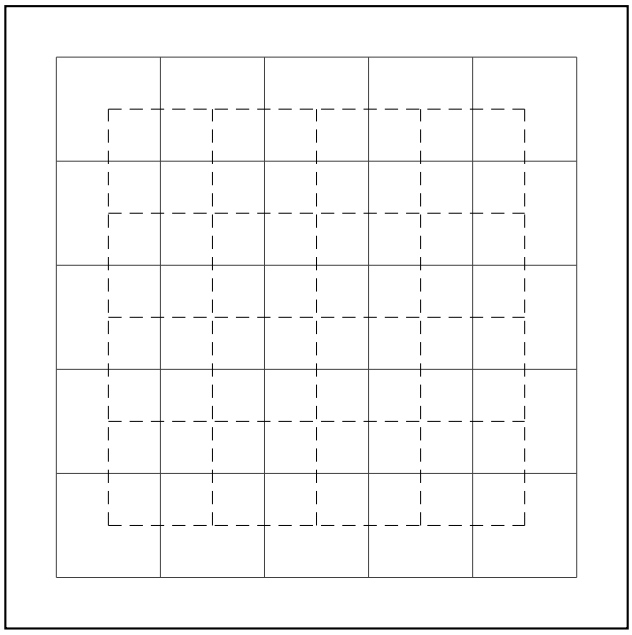


Figure 5.1 The square boxes of the covering Ω are located on two overlapping regular grids. Ω is the union of two coverings Ω_1 and Ω_2, where Ω_1 and Ω_2 are outlined with solid lines and dashed lines respectively.

size one). The set of all the boxes of size L contained in the support of the reference image will be denoted by $\mathcal{B}_r := \{B_1, B_2, ..., B_m\}$, where $m \in \mathbb{N}$. The set of all boxes of size $2L$ contained in the support of the reference image will be denoted by $\mathcal{B}_r^* = \{B_1^*, B_2^*, ..., B_{m^*}^*\}$, where $m^* \in \mathbb{N}$. Similarly, the sets all the boxes of sizes L and $2L$ contained in the support of the *simulated image* will be denoted by $\mathcal{B}_s$ and $\mathcal{B}_s^*$ respectively.

We define the *covering* $\Omega \subset \mathcal{B}_s$ as a set of boxes that covers all the simulated image support except for a strip of width $L/2$ around the support edge (Figure 5.1). More precisely, the boxes in Ω belong to two overlapping regular grids. The first grid, outlined by solid gray lines in Figure 5.1, starts at a distance of $L/2$ from the edges. The second grid, outlined by dashed lines, is shifted by $L/2$ with respect to the first grid. We will write $\Omega = \{B_j \in \mathcal{B}_s : j \in J\}$, where $J \subset \{1, ..., m\}$ denotes the set of indices associated to the covering boxes. In the following, the index j will be used to number the boxes in the covering Ω. Also, the total number of boxes contained in the covering Ω will be denoted by N, i.e. $N := |\Omega|$, where $|\dots|$ denotes cardinality.

We associate to each box $B_j \in \Omega$ of the covering a larger box $B_j^* \in \mathcal{B}_s^*$ of size $2L$ having

the same center as B_j. As shown in Figure 5.2, B_j and B_j^* are concentric and $B_j \subset B_j^*$. Boxes in $\mathcal{B}_s$ and $\mathcal{B}_r$ are numbered so that concentric boxes have the same index. It is stressed that the CPMS works by correcting the image in a suitably chosen box B_j and that this correction process takes into account the current state of the image in the larger box B_j^*. Consequently, the CPMS needs to be able to associate a larger box B_j^* to each box B_j, which is the reason why the covering Ω does not actually cover the whole image support. Indeed, to each box $B_j \in \Omega$ located on the edge of Ω, one associates a larger box B_j^* that overlaps the strip of width $L/2$ around the image support.

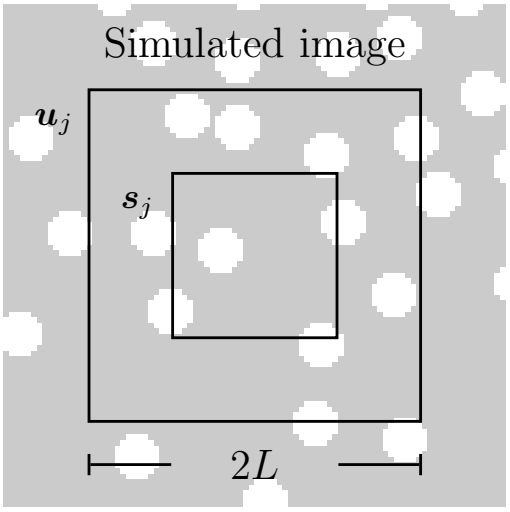


Figure 5.2 The box B_j^* of size $2L$ corresponds to the state $\boldsymbol{u}_j$. The smaller box B_j of size L corresponds to the state $\boldsymbol{s}_j = \boldsymbol{s}(\boldsymbol{u}_j)$.

A *state* is defined as a vector whose components are the pixel values found in a given image region, called the state *support*. We use two types of states that are distinguished by the size of their support (Figure 5.2). States associated to a box of size L will be denoted by $\boldsymbol{s}$ and will be called $\boldsymbol{s}$-states. States associated to a box of size $2L$ will be denoted by $\boldsymbol{u}$ and will be called $\boldsymbol{u}$-states. The set of all distinct $\boldsymbol{s}$-states found in the reference image, i.e. the distinct states associated to the boxes in $\mathcal{B}_r$, will be denoted by $\mathcal{S}_r$. Similarly, the set of all distinct $\boldsymbol{u}$-states associated to $\mathcal{B}_r^*$ will be denoted by $\mathcal{U}_r$. The $\boldsymbol{s}$-state and the $\boldsymbol{u}$-state associated to B_j and B_j^* will be denoted by $\boldsymbol{s}_j$ and $\boldsymbol{u}_j$ respectively. The states $\boldsymbol{s}_j$ and $\boldsymbol{u}_j$ are related by a function $\boldsymbol{s}$ defined by $\boldsymbol{s}(\boldsymbol{u}_j) := \boldsymbol{s}_j$ for all $j \in J$.

Objective functions and local correction method

The functions $E_{\rm pc}(\boldsymbol{x})$, $E_{\rm hd}(\boldsymbol{x})$ and $E_{\rm stat}(\boldsymbol{x})$ are of the form

$$\begin{aligned} E_{\rm pc}(\boldsymbol{x}) &= \sum_{j\in J} e_{\rm pc}(j) \\ E_{\rm hd}(\boldsymbol{x}) &= \sum_{j\in J} e_{\rm hd}(j) \\ E_{\rm stat}(\boldsymbol{x}) &= \sum_{j\in J} e_{\rm stat}(j), \end{aligned} \tag{5.4}$$

where the local errors $e_{\rm pc}$, $e_{\rm hd}$ and $e_{\rm stat}$ (to be defined in the sections 5.3.2 and 5.3.3) quantify respectively pattern-conformity, respect of hard-data and respect of the target histogram within box $B_j \in \Omega$. It follows from (5.2) and (5.4) that

$$E(\boldsymbol{x}) = \sum_{j\in J} e(j), \tag{5.5}$$

where $e(j)$ is a *local error* defined by

$$e(j) := e_{\rm pc}(j) + \mu_1\, e_{\rm hd}(j) + \mu_2\, e_{\rm stat}(j). \tag{5.6}$$

We decrease the local errors iteratively until an acceptable solution $\boldsymbol{x}$ is obtained. Each iteration has three steps:

I) *Selection of the correction location*: choose randomly a box $B_j \in \Omega$, using for each box a probability proportional to the local error $e(j)$ (Figure 5.3a);

II) *Selection of a suitable correction state*: substitute the state $\boldsymbol{s}_j$ for a reference image state $\boldsymbol{s}(\boldsymbol{u}_k) \in \mathcal{S}_r$, where the state $\boldsymbol{u}_k \in \mathcal{U}_r$ is chosen to fit to the patterns found in the large box B_j^*, to respect the local hard data in B_j and to help correct the local-mean histogram (Figures 5.3b-c);

III) *Update of local errors*: update the local errors $e(\ell)$ for the five adjacent boxes $B_\ell \in \Omega$ contained in B_j^*, which are modified by the substitution (Figure 5.3d).

This iteration is stopped if the constraints of problem (5.1) are satisfied and if the last iterates of $E_{\rm pc}(\boldsymbol{x})$ behave like a stationary time-series (see section 5.7.2). The choice of $\boldsymbol{s}(\boldsymbol{u}_k)$ in step II will be discussed in section 5.4. In the following, the index k will always be

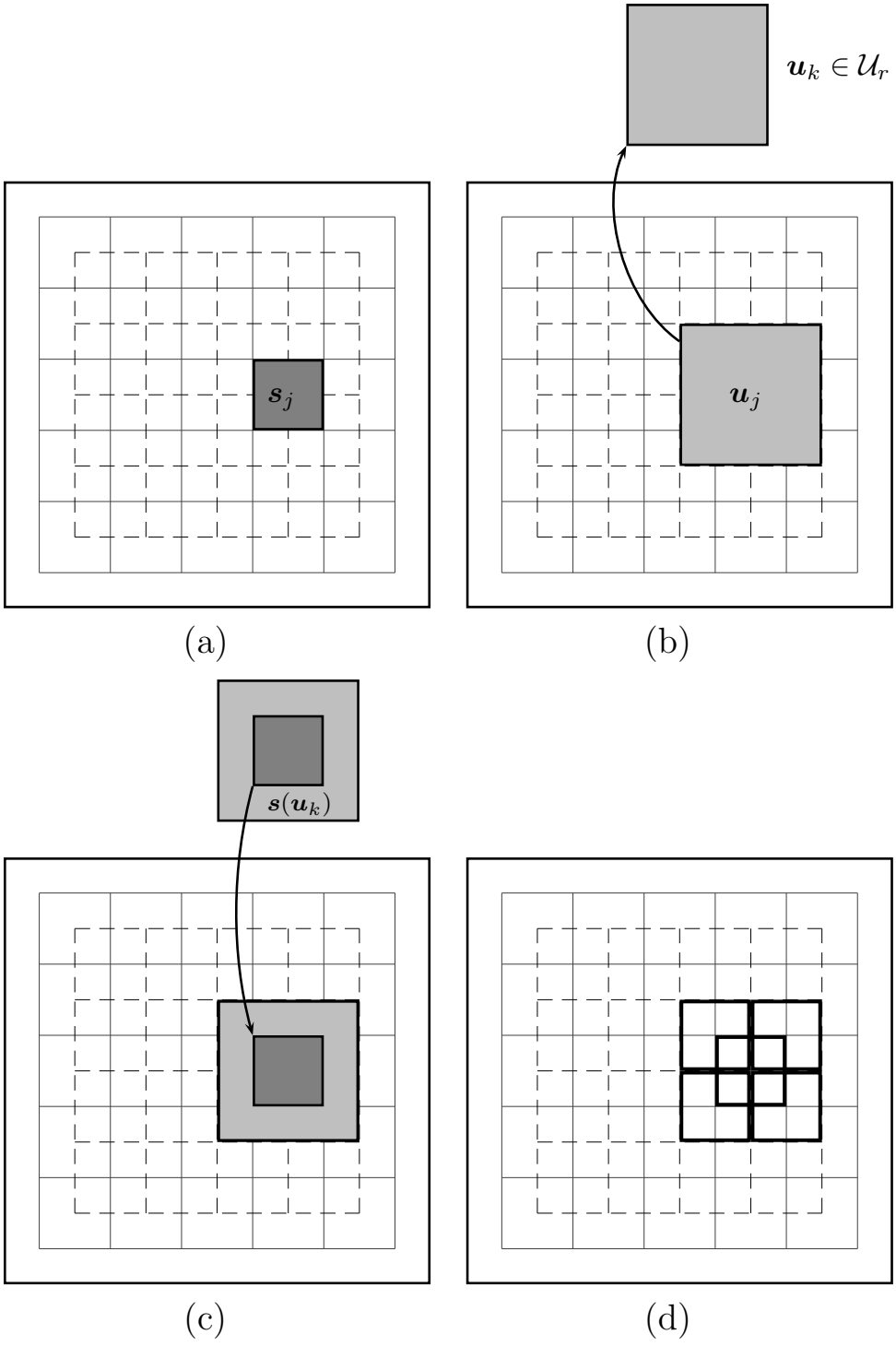


Figure 5.3 (a) The box B_j, in dark gray, selected for correction. (b) Selection of a state $\boldsymbol{u}_k \in \mathcal{U}_r$ based on three criteria: to fit the patterns found in the large box B_j^*, in light gray; to respect the local hard data in B_j; to improve the local-mean histogram. (c) Substitution of the state $\boldsymbol{s}_j$ for the central part of $\boldsymbol{u}_k$, which is the state $\boldsymbol{s}(\boldsymbol{u}_k)$. (d) Update the local errors for the five covering boxes B_k contained in B_j^*.

associated to the correction state $\boldsymbol{u}_k$ chosen in the reference image.

This improvement scheme is *greedy* in the sense that step I modifies the image in the boxes where the local error is large. Our preference for a random choice of the box to correct, by opposition to a deterministic choice (e.g. pick a box for which $e(j)$ is maximal), is motivated by the fact that a random choice reduces the probability of occurrence of periodic oscillations between several boxes during the iteration process. The correction performed in step II either decreases or conserves the local error $e(j)$. However, this correction may cause the local errors of the other four boxes contained in B_j^* to either decrease or increase.

Using the covering Ω instead of all the possible boxes contained in the simulated image support, i.e. in $\mathcal{B}_s$, reduces significantly the computing time because $|\Omega| \ll |\mathcal{B}_s|$. Moreover, the geometry of Ω helps reducing pattern-conformity errors. The covering Ω is the union of two subsets Ω_1 and Ω_2, i.e. $\Omega = \Omega_1 \cup \Omega_2$, where Ω_1 and Ω_2 contain the boxes outlined in Figure 5.1 with solid lines and dashed lines respectively. On the one hand, the boxes of Ω_1 (or Ω_2) form a checkerboard of adjacent and disjoint boxes. On the other hand, a box in Ω_1 typically overlaps four adjacent boxes of Ω_2 that form a square of size $2L$, as illustrated in Figure 5.3d. Conversely, a box in Ω_2 typically overlaps four adjacent boxes of Ω_1 that form a box of size $2L$. This overlap of the coverings Ω_1 and Ω_2 helps capturing the pattern-conformity errors that can occur at the frontier of any pair of adjacent boxes.

To complete the description of our method, we must define the following elements: the functions e_{pc}, e_{hd}, e_{stat}; the step II of the local correction method; the coefficients μ_1, μ_2. These definitions are the topic of the next sections.

5.3 Definition of local errors

5.3.1 Metrics

Several metrics will be used to quantify the similarity between different states. The state type is indicated by a bold symbol $\boldsymbol{y} \in \{\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{z}\}$, where the state type $\boldsymbol{z}$ will be defined in section 5.3.2. The distance between two states $\boldsymbol{y}_j$ and $\boldsymbol{y}_k$ of the same type is defined as

$$d_{\boldsymbol{y}}(\boldsymbol{y}_j, \boldsymbol{y}_k) := \sum_{p=1}^{Q_{\boldsymbol{y}}} w_{\boldsymbol{y}}(p) \left| y_j(p) - y_k(p) \right|, \tag{5.7}$$

where $y_j(p)$ denotes the p^{th} component of the state $\boldsymbol{y}_j$, $Q_{\boldsymbol{y}}$ is the dimension of a state of type $\boldsymbol{y}$ and $w_{\boldsymbol{y}}(p)$ is the weight attributed to the p^{th} pixel for a state of type $\boldsymbol{y}$. Weights depend on the state type.

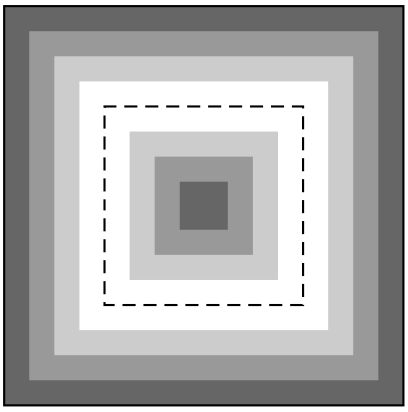


Figure 5.4 Spatial distribution of the weights $w_{\boldsymbol{u}}(p)$. Dark gray and white correspond to the weight values c and $4c$ respectively, where c is a normalization constant chosen so that $\sum_{p=1}^{N_{\boldsymbol{u}}} w_{\boldsymbol{u}}(p) = 1$. For the intermediate gray tones, the weights vary linearly between c and $4c$ in the horizontal (or vertical) direction. The dashed line is the frontier between existing data and patched data, i.e. the outer square and the dashed square are the boundaries of the boxes B_j^* and B_j respectively.

5.3.2 Errors associated to pattern-conformity and hard-data

The pattern-conformity error $e_{\text{pc}}(j)$ is defined as the minimal distance between $\boldsymbol{s}_j$ and the set $\mathcal{S}_r$, i.e.

$$e_{\text{pc}}(j) := \min_{\boldsymbol{s} \in \mathcal{S}_r} d_{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{s}_j), \tag{5.8}$$

where the weights $w_{\boldsymbol{s}}(p)$ used in (5.7) are constant and equal to $1/Q_{\boldsymbol{s}}$, with $Q_{\boldsymbol{s}} = L^2$. It follows that $e_{\text{pc}}(j) \in [0, 1]$ for any state $\boldsymbol{s}_j$, assuming that each pixel value belongs to $\{0, 1\}$. Note that the presence of the minimum in the definition (5.8) implies that the function $e_{\text{pc}}(j)$ is not differentiable with respect to $\boldsymbol{s}_j$ for all $\boldsymbol{s}_j \in \mathbb{R}^{L^2}$. Consequently, the objective function $E_{\text{pc}}(\boldsymbol{x})$ of the original optimization problem (5.1), which is obtained by summing the $e_{\text{pc}}(j)$s for all $j \in J$ (first line of (5.4)), is not differentiable with respect to $\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^Q$.

The subset of hard data found in the box B_j will be called the *local hard data* and will

be denoted by

$$\boldsymbol{z}_j := (z_j(1), z_j(2), \ldots, z_j(Q_j)),$$

where $z_j(p)$ is the p^{th} conditioning pixel value and $Q_j \in \mathbb{N}$ is the number of hard data found in B_j. Define $\boldsymbol{z}(\boldsymbol{s}_j) \in \{0,1\}^{Q_j}$ as an array whose components are the pixel values of $\boldsymbol{s}_j$ at the locations of the local hard-data $\boldsymbol{z}_j$, assuming the same ordering for the components of $\boldsymbol{z}(\boldsymbol{s}_j)$ and $\boldsymbol{z}_j$. We define the local hard-data error as

$$e_{\text{hd}}(j) := d_{\boldsymbol{z}}(\boldsymbol{z}_j, \boldsymbol{z}(\boldsymbol{s}_j)), \tag{5.9}$$

where the weights $w_{\boldsymbol{z}}(p)$ used in (5.7) are equal to one.

5.3.3 Error associated to the local-mean histogram

The CPMS also aims at controlling the local-mean histogram of the simulations. We may want to either simulate images that reproduce the reference image statistical properties, or else simulate images that differ statistically from the reference image. The local-mean associated to the box B_j, denoted by $\phi(\boldsymbol{s}_j)$, is defined as the arithmetic average of the components of $\boldsymbol{s}_j$. It is the histogram of the local-mean $\phi(\boldsymbol{s}_j)$ that will be controlled. If $M \geq 2$ denotes the number of histogram bins (a user-selected integer parameter), then the local-mean histogram absolute frequencies are defined as

$$n(i) = \left|\{j \in J : \varphi_{i-1} \leq \phi(\boldsymbol{s}_j) < \varphi_i\}\right|, \tag{5.10}$$

$i = 1, 2, \ldots, M$, where J is the set of box indices associated to the covering Ω (Figure 5.1) and the φ_is are the histogram bin boundaries. The first and last bin boundaries are always fixed to $\varphi_0 = -\infty$ and $\varphi_M = +\infty$ respectively. In the following, the index i will be used to number the histogram bins. The bin relative frequencies are given by $f(i) = \frac{n(i)}{N}$, $i = 1, 2, \ldots, M$, where $N = \sum_{i=1}^{M} n(i) = |\Omega|$ is the number of boxes in the covering Ω. Our objective is to simulate images that satisfy

$$f(i) \simeq p_t(i), \tag{5.11}$$

$i = 1, \ldots, M$, where the parameters $p_t(i)$ are user-defined *target probabilities*. We define the relative frequency errors as

$$\epsilon(i) := f(i) - p_t(i), \tag{5.12}$$

$i \in \{1, ..., M\}$. The local-mean bin probabilities of the reference image are defined as

$$p_r(i) := \mathrm{P}\left[\varphi_{i-1} \leq \phi(\boldsymbol{S}_r) < \varphi_i\right], \tag{5.13}$$

where the random vector $\boldsymbol{S}_r \in \mathcal{S}_r$ is a state picked randomly in the reference image, using a uniform probability distribution for its support location. It is stressed that $p_t(i)$ may differ from $p_r(i)$ for any i, which allows for simulated images that differ statistically from the reference image.

We designed the *local statistical error* $e_{\mathrm{stat}}(j)$ to penalize the boxes $B_j \in \Omega$ that belong to overpopulated histogram bins. If B_j belongs to an underpopulated bin, then $e_{\mathrm{stat}}(j) := 0$, otherwise $e_{\mathrm{stat}}(j)$ is positive and proportional to the relative frequency error $\epsilon(i)$. In short, $e_{\mathrm{stat}}(j)$ is defined as

$$e_{\mathrm{stat}}(j) := \begin{cases} \epsilon(i(\boldsymbol{s}_j)) & \text{if } \epsilon(i(\boldsymbol{s}_j)) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{5.14}$$

where $i(\boldsymbol{s}_j)$ denotes the local-mean bin index of $\boldsymbol{s}_j$, which is defined implicitly by

$$\varphi_{i(\boldsymbol{s}_j)-1} \leq \phi(\boldsymbol{s}_j) < \varphi_{i(\boldsymbol{s}_j)}. \tag{5.15}$$

It follows from (5.12) and (5.14) that $e_{\mathrm{stat}}(j) \in [0, 1]$. The errors $e_{\mathrm{stat}}(j)$ must be updated at each simulation step.

To test if a simulated image is statistically compatible with the target probabilities, we use the chi-square test, which is based on the statistic

$$\begin{aligned} D^2 &= \sum_{i=1}^{M} \frac{(n(i) - N p_t(i))^2}{N p_t(i)} \\ &= N \sum_{i=1}^{M} \frac{\epsilon^2(i)}{p_t(i)}, \end{aligned} \tag{5.16}$$

where the second line of (5.16) follows from $n(i) = N\, f(i)$ and (5.12). Assuming that D^2

follows a chi-square probability law with $M-1$ degrees of freedom, i.e. $D^2 \sim \chi^2_{M-1}$, the hypothesis that the simulation histogram is compatible with the target probabilities is rejected at the significance level α if $D^2 > \chi^2_{\alpha,M-1}$, where the threshold $\chi^2_{\alpha,n}$ is defined by $\mathrm{P}\left[X \leq \chi^2_{\alpha,n}\right] = 1-\alpha$ for a random variable $X \sim \chi^2_n$. In our simulations, we use $\alpha = 50\%$ and we use the significance threshold $R^2 := \chi^2_{0.5,\,M-1}$.

5.4 Choice of the corrective substitution state

5.4.1 Selection criteria

In step II of the local correction method (section 5.2.2), we substitute the state $\boldsymbol{s}_j$ for the state $\boldsymbol{s}(\boldsymbol{u}_k)$, where $\boldsymbol{u}_k$ is chosen to fit to the patterns found in the large box B_j^*, to respect the local hard data in B_j and to help correct the local-mean histogram. To achieve a balance between these three criteria, we choose the state $\boldsymbol{u}_k$ in the set U_j defined by

$$U_j := \underset{\boldsymbol{u}\in\mathcal{U}_r}{\operatorname{argmin}} \Big[d_{\boldsymbol{u}}\left(\boldsymbol{u}, \boldsymbol{u}_j\right) + d_{\boldsymbol{z}}\left(\boldsymbol{z}_j, \boldsymbol{z}\left(\boldsymbol{s}(\boldsymbol{u})\right)\right) + \gamma\, \epsilon(i(\boldsymbol{s}(\boldsymbol{u}))) \Big], \tag{5.17}$$

where γ is a penalty coefficient. If U_j contains several states, then $\boldsymbol{u}_k$ is selected randomly in U_j with a probability that is proportional to its number of replicas in the reference image.

The pattern-matching error in the large box B_j^* is represented in (5.17) by the distance $d_{\boldsymbol{u}}\left(\boldsymbol{u}, \boldsymbol{u}_j\right)$, for which the weights $w_{\boldsymbol{u}}(p)$ have a spatial distribution designed to favor image patterns continuity. Indeed, patching a state into a box of the simulated image can create pattern discontinuities at the border between new data and existing data (border of $\boldsymbol{s}(\boldsymbol{u}_k)$ in Figure 5.3d). To favor the selection of a state $\boldsymbol{s}(\boldsymbol{u}_k)$ that fits well at this border, we give a larger weight to the pixels that are close to this border, (El Ouassini, Saucier, Marcotte and Favis, 2008; Faucher, Saucier and Marcotte, 2012), as shown in Figure 5.4. More precisely, weights decrease linearly with the distance to the border. The weights $w_{\boldsymbol{u}}(p)$ are normalized, i.e. $\sum_{p=1}^{N_{\boldsymbol{u}}} w_{\boldsymbol{u}}(p) = 1$, which implies that $d_{\boldsymbol{u}}\left(\boldsymbol{u}, \boldsymbol{u}_j\right) \in [0,1]$ for any $\boldsymbol{u} \in \mathcal{U}_r$. We compared simulations results obtained with several scenarios for the spatial distribution of weights: uniform on the box B_j; uniform on the larger box B_j^*; weights concentrated at the border (i.e. gray pixels in Figure 5.4 are set to 0); and the selected distribution displayed in Figure 5.4. We found that the latter weight distribution gives the best results, i.e. it leads to the smallest

values of the global pattern-conformity error $E_{\rm pc}(\boldsymbol{x})$.

The error associated to hard data, which is given by $d_{\boldsymbol{z}}\left(\boldsymbol{z}_j, \boldsymbol{z}\left(\boldsymbol{s}(\boldsymbol{u})\right)\right)$ in (5.17), is greater or equal to one if at least one hard data is not respected in B_j (because $w_{\boldsymbol{z}}(p) = 1, \forall p$). It follows that $d_{\boldsymbol{z}}\left(\boldsymbol{z}_j, \boldsymbol{z}\left(\boldsymbol{s}(\boldsymbol{u})\right)\right) \geq d_{\boldsymbol{u}}\left(\boldsymbol{u}, \boldsymbol{u}_j\right)$ if hard data are not all respected in B_j, which gives priority to hard data correction. This priority ensures that as soon as a box B_j is visited for correction, one of the states $\boldsymbol{s} \in \mathcal{S}_r$ that fits best to the local hard data is patched to B_j and the local hard-data error is minimized, i.e. typically set to zero.

In (5.17), the role of the term $d_{\boldsymbol{z}}(\boldsymbol{z}_j, \boldsymbol{z}(\boldsymbol{s}(\boldsymbol{u})))$ is to limit the choice of $\boldsymbol{u}_k \in \mathcal{U}_r$ to a state that respects the local hard data $\boldsymbol{z}_j$. If $\boldsymbol{z}_j$ can be respected by at least one state $\boldsymbol{u}_k \in \mathcal{U}_r$ in the reference image, which is typically the case, then equation (5.17) takes the simplified form

$$U_j := \underset{\boldsymbol{u} \in \mathcal{U}_r(\boldsymbol{z}_j)}{\operatorname{argmin}} \Big(d_{\boldsymbol{u}}\left(\boldsymbol{u}, \boldsymbol{u}_j\right) \, + \, \gamma \, \epsilon(i(\boldsymbol{s}(\boldsymbol{u}))) \Big), \tag{5.18}$$

where $\mathcal{U}_r(\boldsymbol{z}_j)$ denotes the set of all the possible states $\boldsymbol{u} \in \mathcal{U}_r$ that respect the local hard data $\boldsymbol{z}_j$. The value of γ must be chosen to correct the histogram if it is needed (i.e. if $D^2 > R^2$), and to avoid a histogram drift if the histogram is acceptable (i.e. if $D^2 \leq R^2$). The choice of γ is discussed in the next section.

5.4.2 Choice of a suitable coefficient γ

At the threshold of acceptability, i.e. $D^2 = R^2$, the fluctuations of the relative frequency errors $\epsilon(i) = f(i) - p_t(i)$ can be characterized by the standard deviations σ_i defined by

$$\begin{aligned} \sigma_i^2 &= \mathrm{E}\left[\epsilon^2(i) \,\middle|\, D^2 = R^2\right] \\ &\approx \frac{p_t(i)\Big(1 - p_t(i)\Big)}{N} \frac{R^2}{M-1}, \end{aligned} \tag{5.19}$$

where the second line of (5.19) is derived in appendix 5.A. It can be shown (appendix 5.B) that if $D^2 > R^2$, then there exists at least one pair of indices (i, ℓ) such that $\epsilon(i) - \epsilon(\ell) > \sigma_\epsilon$, where

$$\sigma_\epsilon := \min_{i \in \{1,\ldots,M\}} \sigma_i. \tag{5.20}$$

For any pair (i, ℓ) such that $\epsilon(i) - \epsilon(\ell) > \sigma_\epsilon$, we will say that the bin i is *significantly overpopulated* with respect to the bin ℓ. The set of all pairs (i, ℓ) for which bin i is significantly overpopulated with respect to bin ℓ will be denoted by G, i.e.

$$G := \Big\{(i, \ell) \in \{1, \dots, M\}^2 : \epsilon(i) - \epsilon(\ell) > \sigma_\epsilon\Big\}. \tag{5.21}$$

To determine the effect of γ on the local-mean of the state $\boldsymbol{u}_k$ selected via (5.18), we can split the set $\mathcal{U}_r(\boldsymbol{z}_j)$ into a partition of M subsets $C_i(\boldsymbol{z}_j)$, $i \in \{1, ..., M\}$, that correspond to the M histogram bins. The *classes* $C_i(\boldsymbol{z}_j)$ are defined by

$$C_i(\boldsymbol{z}_j) := \{\boldsymbol{u} \in \mathcal{U}_r(\boldsymbol{z}_j) : \varphi_{i-1} \leq \phi(\boldsymbol{s}(\boldsymbol{u})) < \varphi_i\}, \tag{5.22}$$

$i \in \{1, ..., M\}$, and they satisfy $\mathcal{U}_r(\boldsymbol{z}_j) = \bigcup_{i=1}^{M} C_i(\boldsymbol{z}_j)$ and $C_i(\boldsymbol{z}_j) \cap C_\ell(\boldsymbol{z}_j) = \emptyset$ for $i \neq \ell$. It follows from this partition that

$$\min_{\boldsymbol{u} \in \mathcal{U}_r(\boldsymbol{z}_j)} \Big(d_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{u}_j) + \gamma\, \epsilon(i(\boldsymbol{s}(\boldsymbol{u})))\Big) = \min_{i \in \{1,\dots,M\}} \min_{\boldsymbol{u} \in C_i(\boldsymbol{z}_j)} \Big(d_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{u}_j) + \gamma\, \epsilon(i(\boldsymbol{s}(\boldsymbol{u})))\Big) \tag{5.23}$$

and therefore the histogram bin-index $i(\boldsymbol{s}(\boldsymbol{u}_k))$ satisfies

$$i(\boldsymbol{s}(\boldsymbol{u}_k)) \in \operatorname*{argmin}_{i \in \{1,\dots,M\}} \min_{\boldsymbol{u} \in C_i(\boldsymbol{z}_j)} \Big(d_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{u}_j) + \gamma\, \epsilon(i(\boldsymbol{s}(\boldsymbol{u})))\Big). \tag{5.24}$$

Noticing that $\epsilon(i(\boldsymbol{s}(\boldsymbol{u}))) = \epsilon(i)$ for all $\boldsymbol{u} \in C_i(\boldsymbol{z}_j)$, equation (5.24) becomes

$$i(\boldsymbol{s}(\boldsymbol{u}_k)) \in \operatorname*{argmin}_{i \in \{1,\dots,M\}} \left(\delta_j(i) + \gamma\, \epsilon(i)\right), \tag{5.25}$$

where $\delta_j(i)$ is the distance between $\boldsymbol{u}_j$ and the set $C_i(\boldsymbol{z}_j)$, i.e.

$$\delta_j(i) := \min_{\boldsymbol{u} \in C_i(\boldsymbol{z}_j)} d_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{u}_j). \tag{5.26}$$

It follows from (5.25) that the bin of index i is excluded from the solutions if there exists

at least one bin of index ℓ such that

$$\delta_j(\ell) + \gamma\ \epsilon(\ell) < \delta_j(i) + \gamma\ \epsilon(i). \tag{5.27}$$

We want to select a bin that will improve the histogram if it is needed. An improvement is needed if $D^2 > R^2$, in which case there exists at least one bin that is significantly overpopulated with respect to other bins. To improve the histogram, we want to avoid choosing such a bin. In other words, we want to exclude from the solutions any bin i that is significantly overpopulated with respect to another bin $\ell \neq i$. More precisely, we need (5.27) to be satisfied for all (i, ℓ) in G, where the set G defined in (5.21), which implies that

$$\gamma > \frac{\delta_j(\ell) - \delta_j(i)}{\epsilon(i) - \epsilon(\ell)} \text{ for all } (i,\ \ell) \in G. \tag{5.28}$$

Since the CPMS visits in priority locations where statistical errors are observed, i.e. where $\epsilon(i(\boldsymbol{s}_j)) > 0$, using a value of γ that satisfies (5.28) ensures the depletion of overpopulated bins.

To satisfy (5.28), it is sufficient to choose the lower bound

$$\gamma \ > \ \frac{\Delta_j}{\sigma_\epsilon}, \tag{5.29}$$

where the range Δ_j is defined as $\Delta_j := \max_{i\in\{1,\ldots,M\}} \delta_j(i) - \min_{i\in\{1,\ldots,M\}} \delta_j(i)$. The lower bound on γ defined by (5.29), i.e. Δ_j/σ_ϵ, depends on the index j of the box chosen randomly for correction. Respecting (5.29) ensures the choice of a state that improves the histogram. We propose to choose the value of γ so that the probability of improving the histogram at each simulation step is at least $1/2$, i.e. we choose

$$\mathrm{P}\left[\gamma > \frac{\Delta_j}{\sigma_\epsilon}\right] = 1/2. \tag{5.30}$$

Denoting by Δ the median of the values of Δ_j in the previous N simulation steps, (5.30) finally leads to

$$\gamma = \frac{\Delta}{\sigma_\epsilon}. \tag{5.31}$$

If the local-mean histogram is unacceptable, then the choice (5.31) ensures that the CPMS is more likely to correct the histogram than to correct pattern-conformity errors.

Let us denote by $\boldsymbol{x}_n$ the simulated image obtained after n steps of the iterative correction process described in section 5.2.2, $n \in \{0, 1, 2, ...\}$. In our simulations, we observed that the value of γ has a significant effect on the number of iterations required to obtain an approximate solution of problem (5.1): firstly, γ influences the minimal number of iterations needed for the statistics D^2 to be maintained below the threshold R^2; secondly, γ also influences the rate at which the conformity error $E_{\mathrm{pc}}(\boldsymbol{x}_n)$ reaches a minimal plateau. The effect of γ on these convergence properties is illustrated in Figure 5.5 for the simulation of images composed of disks presented in section 5.8.3. In Figure 5.5, the average values of D^2 and $E_{\mathrm{pc}}(\boldsymbol{x}_n)$ obtained from 16 independent simulations are displayed as a function of the iteration step n for several different values of γ: we use the values $\gamma_0/10$, γ_0, $10\,\gamma_0$ and $100\,\gamma_0$, where $\gamma_0 := \Delta/\sigma_\epsilon$ is the value recommended in (5.31). The only value of γ that does not allow the simulation to maintain $D^2 \leq R^2$ is $\gamma = \gamma_0/10$, which is too small. For all simulations but the one with $\gamma = \gamma_0/10$, we observe that the largest decrease of $E_{\mathrm{pc}}(\boldsymbol{x}_n)$ occurs during the first N iterations. The first N iterations correspond to a construction phase during which each of the N covering boxes is corrected about once. For $n > N$, the decrease of $E_{\mathrm{pc}}(\boldsymbol{x}_n)$ is slower. In Figure 5.5, we observe that the lowest value of $E_{\mathrm{pc}}(\boldsymbol{x}_n)$ is obtained with $\gamma = \gamma_0/10$, for which the statistics D^2 significantly exceeds the acceptable threshold $R^2 = 0.46$. The value $\gamma = \gamma_0$ gives the best solution in the sense that it produces the lowest value of $E_{\mathrm{pc}}(\boldsymbol{x}_n)$ among the solutions that satisfies the statistical criterion $D^2 < R^2$.

5.5 Selection of the correction site: penalty coefficient μ_2

For an image for which all hard data are satisfied, i.e. $e_{\mathrm{hd}}(j) = 0$ for all $j \in J$, the local error given by (5.5) takes the form $e(j) = e_{\mathrm{pc}}(j) + \mu_2\, e_{\mathrm{stat}}(j)$. This form shows that the coefficient μ_2 weighs the relative contributions of $e_{\mathrm{pc}}(j)$ and $e_{\mathrm{stat}}(j)$ to the local error $e(j)$. We choose μ_2 so that the two quantities $e_{\mathrm{pc}}(j)$ and $\mu_2\, e_{\mathrm{stat}}(j)$, where $j \in J$, are equal on average for a simulated image such that $D^2 = R^2$. More precisely, we define μ_2 implicitly by the identity

$$\mathrm{E}\left[e_{\mathrm{pc}}(j)\right] = \mu_2\, \mathrm{E}\left[e_{\mathrm{stat}}(j) \,\middle|\, D^2 = R^2\right]. \tag{5.32}$$

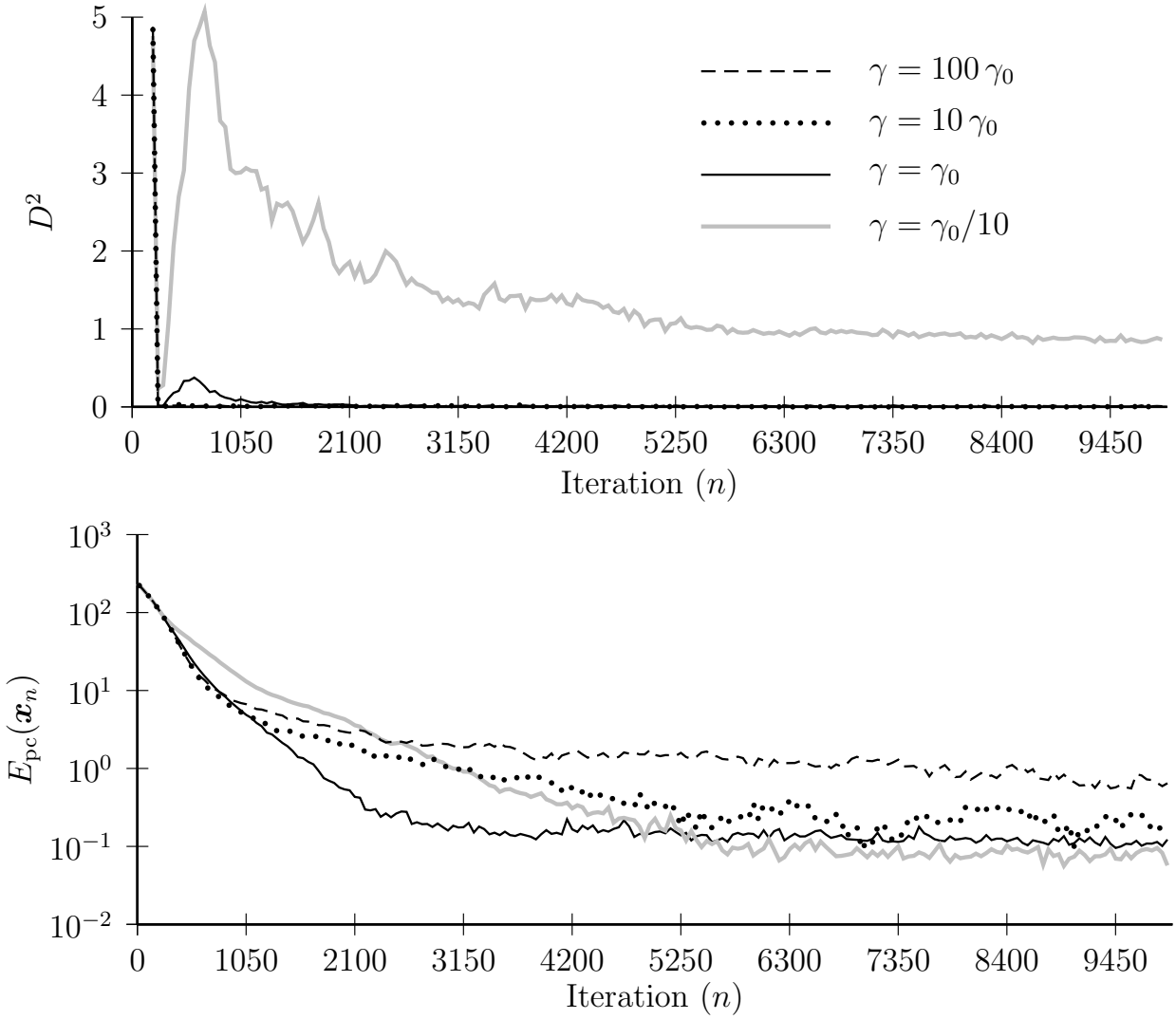


Figure 5.5 Evolution of the statistics D^2 (**top**) and the global pattern-conformity error $E_{\mathrm{pc}}(\boldsymbol{x}_n)$ (**bottom**) for 10000 iterations and four different choices of γ. The number of boxes in the covering is $N = 525$ and the threshold on the chi-square test is $R^2 = 0.46$. The reference image is composed of disks (simulations of images composed of disks are presented in section 5.8.3).

Our motivation for enforcing this equality is to ensure that pattern-matching errors and histogram-matching errors have a comparable influence on the determination of the correction site for images at the threshold of statistical acceptability.

Since $\mathrm{E}\left[e_{\mathrm{stat}}(j) \,\middle|\, D^2 = R^2\right]$ is proportional to R (as shown in appendix 5.C), it follows from the definition (5.32) of μ_2 that for an image having an unacceptable local-mean histogram, we have

$$\mathrm{E}\left[e_{\mathrm{pc}}(j)\right] < \mu_2 \, \mathrm{E}\left[e_{\mathrm{stat}}(j) \middle| D^2 > R^2\right], \tag{5.33}$$

i.e. histogram-matching errors have a larger influence than pattern-matching errors on the determination of the correction site. Conversely, for an image having an acceptable local-mean histogram, we have

$$\mathrm{E}\left[e_{\mathrm{pc}}(j)\right] \geq \mu_2 \, \mathrm{E}\left[e_{\mathrm{stat}}(j) \middle| D^2 \leq R^2\right], \tag{5.34}$$

i.e. histogram-matching errors have a smaller influence than pattern-matching errors on the determination of the correction site.

We estimate the term $\mathrm{E}\left[e_{\mathrm{pc}}(j)\right]$ in (5.32) via the mean pattern-conformity error at the n^{th} simulation step, i.e. we use

$$\mathrm{E}\left[e_{\mathrm{pc}}(j)\right] = \frac{E_{\mathrm{pc}}(\boldsymbol{x})}{N}. \tag{5.35}$$

In appendix 5.C, it is shown that the expected value of the right hand side of (5.32) is given by

$$\mathrm{E}\left[e_{\mathrm{stat}}(j) \,\middle|\, D^2 = R^2\right] = \frac{\beta(M)}{2} \sum_{i=1}^{M} \sigma_i \, p_t(i), \tag{5.36}$$

where the σ_is are given by (5.19) and $\beta(M) \in [\sqrt{2/\pi}, 1]$ is a coefficient defined by equation (5.75). Substituting (5.35) and (5.36) into (5.32) yields

$$\mu_2 = \frac{2\, E_{\mathrm{pc}}(\boldsymbol{x})}{N \; \beta(M) \overline{\sigma}}, \tag{5.37}$$

where $\overline{\sigma}$ is a weighted average of the σ_is defined as

$$\overline{\sigma} := \sum_{i=1}^{M} \sigma_i \; p_t(i). \tag{5.38}$$

Note that the penalty coefficient μ_2 is not constant since it is proportional to $E_{\mathrm{pc}}(\boldsymbol{x})$, which varies with the simulation step.

We observed experimentally that the choice (5.37) allows $E_{\mathrm{pc}}(\boldsymbol{x})$ to reach a minimal plateau fairly rapidly, i.e. within about $8\,N$ iterations, while maintaining $D^2 \leq R^2$. We observed that using a value of μ_2 much smaller than (5.37) can make the CPMS dependent on the image used to initialize the simulation. Let us consider, for instance, the simulation of white disks on a black background (see section 5.8.3 for simulation results). If the simulated image is initialized with white noise, then we found that a very small value of μ_2 can be used without affecting the convergence of the CPMS. However, if the initialization image is all black, then choosing $\mu_2 \ll {}^{2\,E_{\mathrm{pc}}(\boldsymbol{x})}/_{N\ \beta(M)\overline{\sigma}}$ can lead to a slow convergence of the CPMS or even to no convergence. In the latter case, the simulation starts by patching a few white disks on the black background to adjust the local-mean histogram. Then, since μ_2 is very small, the CPMS stops caring about the local statistical error $e_{\mathrm{stat}}(j)$ and keeps visiting the locations around the few white disks where some local pattern-conformity errors exist, i.e. where $e_{\mathrm{pc}}(j) > 0$ (black areas having $e_{\mathrm{pc}}(j) = 0$). This results in a simulated image in which a cluster of white disks either grows slowly from the initial white disks (i.e. slow convergence) or else does not grow at all (i.e. no convergence).

5.6 Choice of the penalty coefficient μ_1

We want to choose μ_1, which weights the hard data error in equation (5.6), so that the improvement scheme gives priority to the correction of the boxes that fail to respect the hard data. The value of μ_1 must be sufficiently large to give priority to the respect of hard-data at any simulation step. If this is the case, then the precise value of μ_1 does not have any significant effect on the simulation because hard-data errors vanish as soon as all hard-data are satisfied. In this spirit, we search for an acceptable lower bound for μ_1.

The boxes that fail to respect the hard data have indices that form a set $\mathcal{J}$ defined as $\mathcal{J} := \{j \in J : e_{\mathrm{hd}}(j) \geq 1\}$. We want the term $\mu_1\, e_{\mathrm{hd}}(j)$ in (5.5) to be greater or equal than

the other two error terms for all $j \in \mathcal{J}$, i.e.

$$\begin{aligned} \mu_1\, e_{\text{hd}}(j) &\geq e_{\text{pc}}(j) + \mu_2\ e_{\text{stat}}(j), \quad \forall j \in \mathcal{J}, \\ \Rightarrow \mu_1 &\geq \frac{e_{\text{pc}}(j) + \mu_2\ e_{\text{stat}}(j)}{e_{\text{hd}}(j)}, \quad \forall j \in \mathcal{J}, \\ \Rightarrow \mu_1 &\geq \max_{j \in \mathcal{J}} \frac{e_{\text{pc}}(j) + \mu_2\ e_{\text{stat}}(j)}{e_{\text{hd}}(j)}. \end{aligned} \tag{5.39}$$

Since

$$\begin{aligned} \max_{j \in \mathcal{J}} \frac{e_{\text{pc}}(j) + \mu_2\ e_{\text{stat}}(j)}{e_{\text{hd}}(j)} &\leq \frac{\max\limits_{j \in \mathcal{J}} \Big(e_{\text{pc}}(j) + \mu_2\ e_{\text{stat}}(j)\Big)}{\min\limits_{j \in \mathcal{J}} e_{\text{hd}}(j)} \\ &\leq \max_{j \in \mathcal{J}} \Big(e_{\text{pc}}(j) + \mu_2\ e_{\text{stat}}(j)\Big) \\ &\leq \max_{j \in \mathcal{J}} e_{\text{pc}}(j) + \mu_2\ \max_{j \in \mathcal{J}} e_{\text{stat}}(j) \\ &\leq 1 + \mu_2, \end{aligned} \tag{5.40}$$

where the last line of (5.40) results from the fact that both $e_{\text{pc}}(j)$ and $e_{\text{stat}}(j)$ are bounded from above by one, it is sufficient to satisfy

$$\mu_1 \geq 1 + \mu_2 \tag{5.41}$$

to satisfy (5.39). Substituting μ_2 in (5.41) for the expression (5.37) yields

$$\mu_1 \geq 1 + \frac{2\, E_{\text{pc}}(\boldsymbol{x})}{N \beta(M) \overline{\sigma}}. \tag{5.42}$$

We want condition (5.42) to be respected at each simulation step. Since the largest value of $E_{\text{pc}}(\boldsymbol{x})$ occurs immediately after the image initialization, we substitute $E_{\text{pc}}(\boldsymbol{x})$ in (5.42) by an upper bound of its expected value after initialization. This expected value is given by $\mathrm{E}\left[E_{\text{pc}}(\boldsymbol{x})\right] = \sum_{j \in I_\Omega} \mathrm{E}\left[e_{\text{pc}}(j)\right]$. Since $\mathrm{E}\left[e_{\text{pc}}(j)\right] = \mathrm{E}\left[\min_{\boldsymbol{s} \in \mathcal{S}_r} d_{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{s}_j)\right] \leq \mathbb{E}\left[d_{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{s}_j)\right]$, then

$$\mathrm{E}\left[E_{\text{pc}}(\boldsymbol{x})\right] \leq N\ \mathbb{E}\left[d_{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{s}_j)\right].$$

Since the image is initialized with a white noise for which each pixel takes the values 0 or 1 with equal probabilities, then $\boldsymbol{s}_j$ is a random vector containing white noise and consequently $\mathbb{E}\left[d_{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{s}_j)\right] = 1/2$, $\forall \boldsymbol{s} \in \mathcal{S}_r$. It follows that $\mathrm{E}\left[E_{\text{pc}}(\boldsymbol{x})\right] \leq \frac{N}{2}$ at the initialization. Substituting

$E_{\mathrm{pc}}(\boldsymbol{x})$ in (5.42) by the upper bound $\frac{N}{2}$ finally yields

$$\mu_1 \;\geq\; 1 + \frac{1}{\beta(M)\overline{\sigma}}\,. \tag{5.43}$$

5.7 Algorithmic considerations

5.7.1 Compatibility of hard-data and reference image

The hard data constraint is typically satisfied after a few iterations if the hard data are compatible with the reference image. In this case, the first steps of the simulation are to substitute the states $\boldsymbol{s}_j$ that contain non-respected hard data for states $\boldsymbol{s}(\boldsymbol{u}_k) \in \mathcal{S}_r$ that satisfy the hard data. If the hard data are not fully compatible with the reference image, i.e. if there are boxes B_j for which no state $\boldsymbol{s}(\boldsymbol{u}_k) \in \mathcal{S}_r$ can satisfy the local hard data $\boldsymbol{z}_j$, then the CPMS cannot honor all the hard data and the simulation stops. As mentioned by Arpat and Caers (2007), this occurs typically if the reference image is not representative of the hard data. In this case, the simplest remedy is to supplement $\mathcal{S}_r$ with new states that can honor all the hard data, which can be done by using a different or a larger reference image. If a larger reference image is not available, it is also possible to generate one from the available reference image using e.g. the patchwork simulation method (Faucher, Saucier and Marcotte, 2012).

5.7.2 Monitoring of the iterative correction process

Let us recall that $\boldsymbol{x}_n$ denotes the n^{th} iteration of the simulation in the iterative correction process described in section 5.2.2. As the iteration runs, we first wait for all the hard data to be respected. The constraint on the hard data is typically the first to be satisfied, and since the term $d_{\boldsymbol{z}}\left(\boldsymbol{z}_j, \boldsymbol{z}\left(\boldsymbol{s}(\boldsymbol{u})\right)\right)$ in (5.17) limits the choice of a state $\boldsymbol{s}(\boldsymbol{u}_k)$ that satisfies the hard data, this constraint remains satisfied for the rest of the simulation.

Since we search for solutions of problem (5.1), in the subsequent iteration steps we store the image $\boldsymbol{x}_n$ as soon as it satisfies the chi-square test. This image is stored as the n^{th}-step best solution and the value of $E_{\mathrm{pc}}(\boldsymbol{x}_n) =: E_{\mathrm{pc}}|_{\mathrm{min}}$ is stored. In the following iterations, for each n such that both conditions $E_{\mathrm{pc}}(\boldsymbol{x}_n) < E_{\mathrm{pc}}|_{\mathrm{min}}$ and $D^2(\boldsymbol{x}_n) \leq R^2$ are satisfied, the image

$\boldsymbol{x}_n$ is stored as the n^{th}-step best solution and $E_{\text{pc}}|_{\text{min}}$ is updated to $E_{\text{pc}}(\boldsymbol{x}_n)$.

We let the simulation continue until the global pattern-conformity error $E_{\text{pc}}(\boldsymbol{x}_n)$ reaches a plateau (Figure 5.5). Since the image is initialized with a white noise, $E_{\text{pc}}(\boldsymbol{x}_n)$ starts with an high value and decreases as n increases. The simulation is stopped if the global pattern-conformity error stops decreasing, i.e. if

$$\overline{E}_{\text{pc}}(n) \geq \overline{E}_{\text{pc}}(n-N), \tag{5.44}$$

where $\overline{E}_{\text{pc}}(n)$ denotes the sliding-window average of the iterates $E_{\text{pc}}(\boldsymbol{x}_n)$, which is defined by $\overline{E}_{\text{pc}}(n) := \frac{1}{N}\sum_{p=n-N+1}^{n} E_{\text{pc}}(\boldsymbol{x}_p)$. We observed experimentally that the stopping criterion (5.44) detects efficiently the simulation step for which the image ceases to evolve. For instance, the simulation with $\gamma = \gamma_0$ for which the evolution of $E_{\text{pc}}(\boldsymbol{x}_n)$ is shown in Figure 5.5 would have stopped after 4725 iterations, which corresponds here to $9N$ iterations.

Note that it is not necessarily possible to obtain an image that satisfies the chi-square test. Indeed, the target probabilities can be given values that are incompatible with the reference image, e.g. one could try to simulate a white image using an all black reference image. In such cases, the simulation is stopped when $E_{\text{pc}}(\boldsymbol{x}_n)$ reaches a plateau (condition (5.44)), and we store as the best image the image for which the chi-square statistics $D^2(\boldsymbol{x}_n)$ is smallest.

5.7.3 Search trees

The part of our method that is most intensive computationally is the search for the set U_j (equation (5.17)) and the update of the values of $e_{\text{pc}}(j)$. The determination of U_j and $e_{\text{pc}}(j)$ requires to scan the sets $\mathcal{U}_r$ and $\mathcal{S}_r$ respectively. As for the PSM (Faucher, Saucier and Marcotte, 2012), the CPMS takes advantage of a tree structure for state storage to speed-up these searches. We used a tree structure called *spill trees* (Liu, Moore, Gray and Yang, 2004). This structure is very efficient to search in high-dimensional spaces such as the spaces of the $\boldsymbol{u}$-states or the $\boldsymbol{s}$-states, which have dimensions of $4L^2$ and L^2 respectively, where $L > 10$. Note that the control of the local-mean histogram requires that one such tree be built for each of the M local-mean classes.

5.8 Simulation results

5.8.1 Reference images

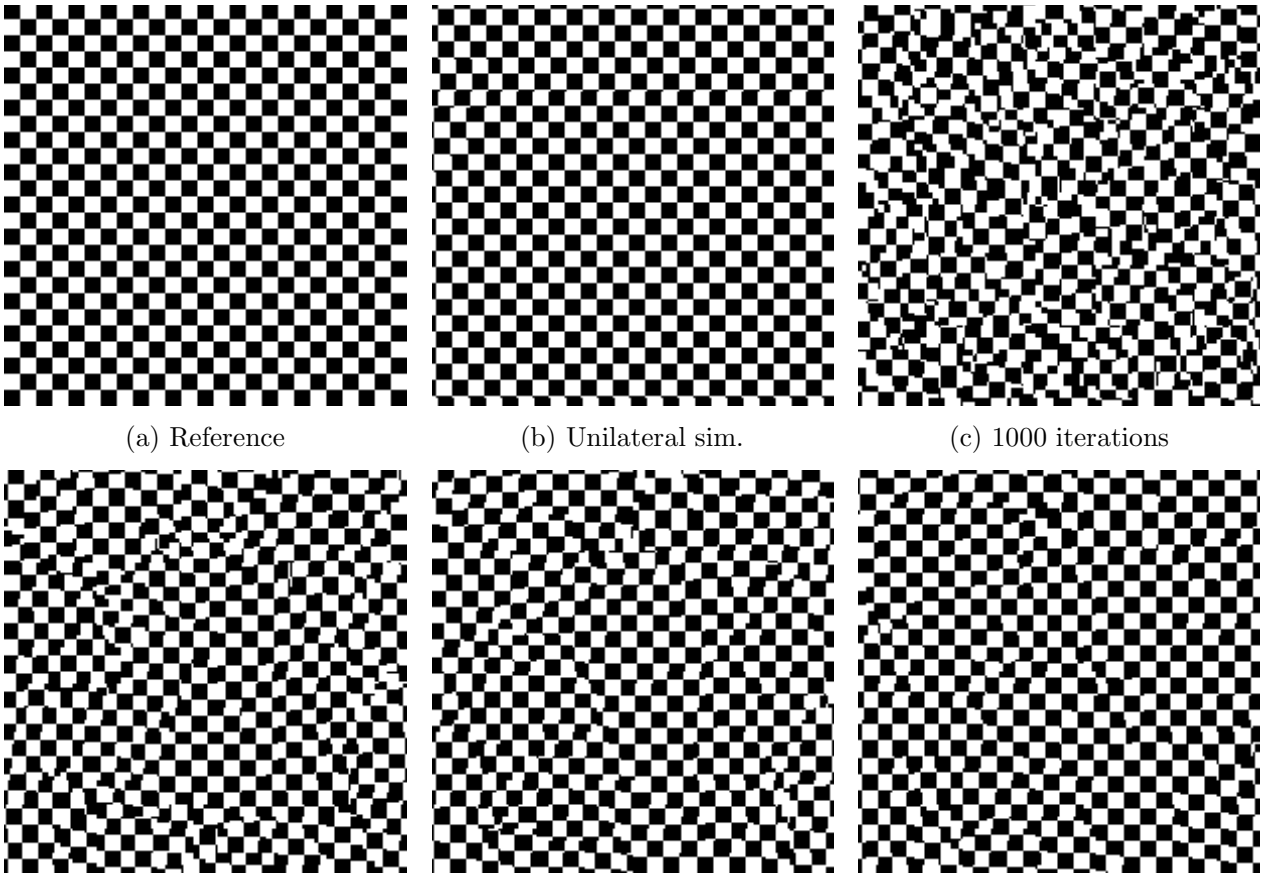
We tested the CPMS with four types of reference images. The first one is a checkerboard image, which allows us to test the ability of the CPMS to reproduce a regular periodic pattern. The second reference image is composed of white disks located randomly on a black background. This image allows us to demonstrate the ability of our method to control the density of disks in simulated images by varying the histogram target probabilities. It also allows us to demonstrate the ability of the CPMS to assemble properly the regular disks that compose these images. The third type of reference image is a scanning electron microscope photography of a polymer blend, which exhibits a larger variety of complex patterns. This image allows us to demonstrate the ability of our method to control the histogram with a higher resolution. The last reference image is composed of a synthetic network of channels. With this image, we examine the effect of the hard data and of the box size L on the simulations.

The values of L considered for these simulations are in the range [6, 30]. All simulated images are of size 200 $\times$ 200 pixels. Since the width and height of the covering Ω must be multiples of L, and taking into account the surrounding strip of width $L/2$, the simulations are performed on images of size $(\left\lceil \frac{200}{L} \right\rceil L + L) \times (\left\lceil \frac{200}{L} \right\rceil L + L)$ pixels. When the simulation is finished, a central square region of size 200 $\times$ 200 pixels is cropped to obtain the final image. All simulated images are initialized with a white noise for which each pixel takes the value 0 or 1 with equal probability.

5.8.2 Checkerboard as a reference image

The checkerboard reference image is composed of squares of size 8 pixels. The box size used for the simulation is $L = 10$. We need L to be greater than the size of the checkerboard square to reproduce its patterns. The simulations based on the checkerboard are not constrained, i.e. there is no hard data and no control of the local-mean (we use $\gamma = 0$ and $\mu_2 = 0$).

We observe in the Figures 5.6c-e that the simulation succeeds in reproducing parts of



(a) Reference (b) Unilateral sim. (c) 1000 iterations

(d) 5000 iterations (e) 10000 iterations (f) 25000 iterations

Figure 5.6 Checkerboard simulations as a function of the number of iterations.

the checkerboard locally but that these parts are not properly aligned with one another. This phenomenon resembles the formation of grains in polycrystalline materials, in which crystallization occurs locally but not globally, thus forming an assembly of crystalline grains. As in polycrystalline materials, the energy of the interface between two grains is high, i.e. the local error e_{pc} is high at the interface between two crystalline regions.

The formation of grains is a consequence of the path used to make corrections, i.e. choosing randomly a location having a large local error at each simulation step. The first simulation steps create many small grains all over the simulated image (Figure 5.6c). Next, small grains merge to create bigger grains (Figure 5.6d). Later, the image stops to evolve significantly and stabilizes in a state in which a few large grains remain (Figures 5.6e-f).

In comparison, checkerboard images simulated with the patchwork simulation method

(Faucher, Saucier and Marcotte, 2012) are perfect, i.e. they do not exhibit grains (Figure 5.6b). Since this unilateral method works by constructing the image from left to right and up to bottom, it can succeed in constructing a periodic pattern indefinitely as long as the initial horizontal and vertical strips are well constructed (Faucher, Saucier and Marcotte, 2012).

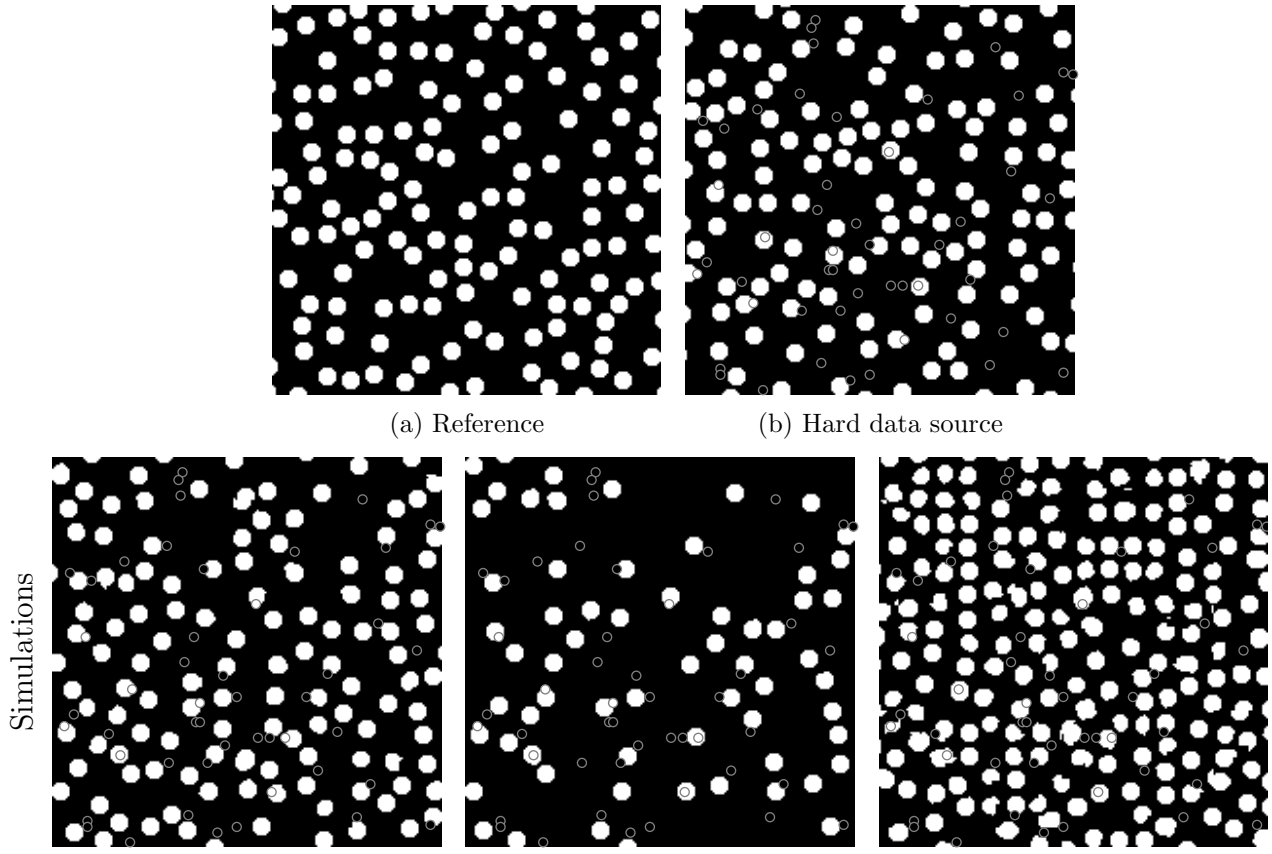
5.8.3 Images composed of disks

The reference image is composed of constant-diameter disjoint white disks located randomly on a uniform black background (Figure 5.7a). The simulations are constrained by 50 hard data sampled randomly from a second image (Figure 5.7b) generated in the same way as the reference image, which helps making the hard data compatible with the reference image.

The box size is $L = 12$ pixels and the disks diameter is 9 pixels. As for the patchwork simulation method (Faucher, Saucier and Marcotte 2012), the optimal box size L is related to the reference image correlation length λ_1. In this case, we observed that the patterns are best reproduced with $L = 12$, which is approximately equal to λ_1. The number of local-mean histogram bins is $M = 2$ and the middle bin boundary is $\varphi_1 \simeq 0.19$, i.e. the states with less than 19% white go to the first bin and the others go to the second bin. We chose the bin boundaries so that the bin-probabilities $p_r(i)$ have similar values, i.e. $p_r(1) \simeq p_r(2) \simeq 1/2$.

In the first simulation (Figure 5.7c), we set $\boldsymbol{p}_t = \boldsymbol{p}_r$. It was found that the CPMS is compatible with the target local-mean histogram, respects the hard data and reproduces correctly the shape of nearly all disks. In the second simulation (Figure 5.7d), we set $p_t(1) > p_r(1)$, i.e. $\boldsymbol{p}_t = (80\%, 20\%)$. This choice of $\boldsymbol{p}_t$ corresponds to 80% of all-black states and should therefore lead to simulations having a lower density of white disks. It was found that the CPMS respects the hard data, is compatible with the target local-mean histogram and reproduces correctly the shape of all disks.

In the third simulation (Figure 5.7e), we set $p_t(1)$ to a small value (i.e. $\boldsymbol{p}_t = (10\%, 90\%)$) to obtain a density of disks higher than the reference image. It was found that the CPMS respects the hard data and is statistically compatible with the target histogram, but the disks morphology is not reproduced as well as in the previous two simulations. Since the reference



(a) Reference (b) Hard data source

(c) $\boldsymbol{p}_t = \boldsymbol{p}_r$ (d) Low disk density (e) High disk density

Figure 5.7 Simulations of images composed of disks. Each hard datum is represented by a small gray circle.

image is poor in patterns having a high density of disks, the CPMS fails to build perfectly shaped disks because it does not have a sufficiently large collection of high-density patterns to choose from.

5.8.4 Simulation based on a polymer blend image

Reproducing the reference image statistical properties

The simulations are based on the reference image displayed on top of Figure 5.8, which was obtained by thresholding to black-and-white a scanning electron microscopy image of a polymer blend. The complex microstructure of the polymer blend is associated with a continuum of values of the local-mean, which makes this type of image well suited for a control of the local-mean histogram with a higher resolution, i.e. using more than two bins. As shown with unilateral patchwork simulations (Faucher, Saucier and Marcotte 2012), using eight bins instead of two results in a more faithful reproduction of the reference image local-mean histogram. In this section, we perform simulations using eight bins, i.e. $M = 8$. All simulations aim at reproducing the reference image statistical properties, i.e. we use $\boldsymbol{p}_t = \boldsymbol{p}_r$, and we compare the simulations obtained with different box sizes L, i.e. $L \in \{6, 12, 20, 30\}$. Note that $L = 12$ corresponds approximately to the polymer blend image correlation length. This can be seen from the variogram displayed in Figure 5.11, which reaches a constant plateau around $L = 12$.

In the first series of simulations, called unconstrained simulations and displayed in the first row of Figure 5.8, we used the whole polymer blend image as the reference image and no hard data were introduced. In the second series of simulations, called the constrained simulations and displayed in the second row of Figure 5.8, we used only the left-hand side half of the polymer blend image as the reference image and we selected randomly 50 hard data from the right-hand side half of the same image. This is a way to create hard data that are independent from yet compatible with the reference image.

To compare the simulated images with the original scan, we measured the *cord length distribution* (CLD) and the variograms (Figures 5.9, 5.10 and 5.11). Cords are the black line segments of thickness one pixel that are found at the intersection of the image with a straight line. The cord lengths were measured for all lines in the horizontal and vertical directions.

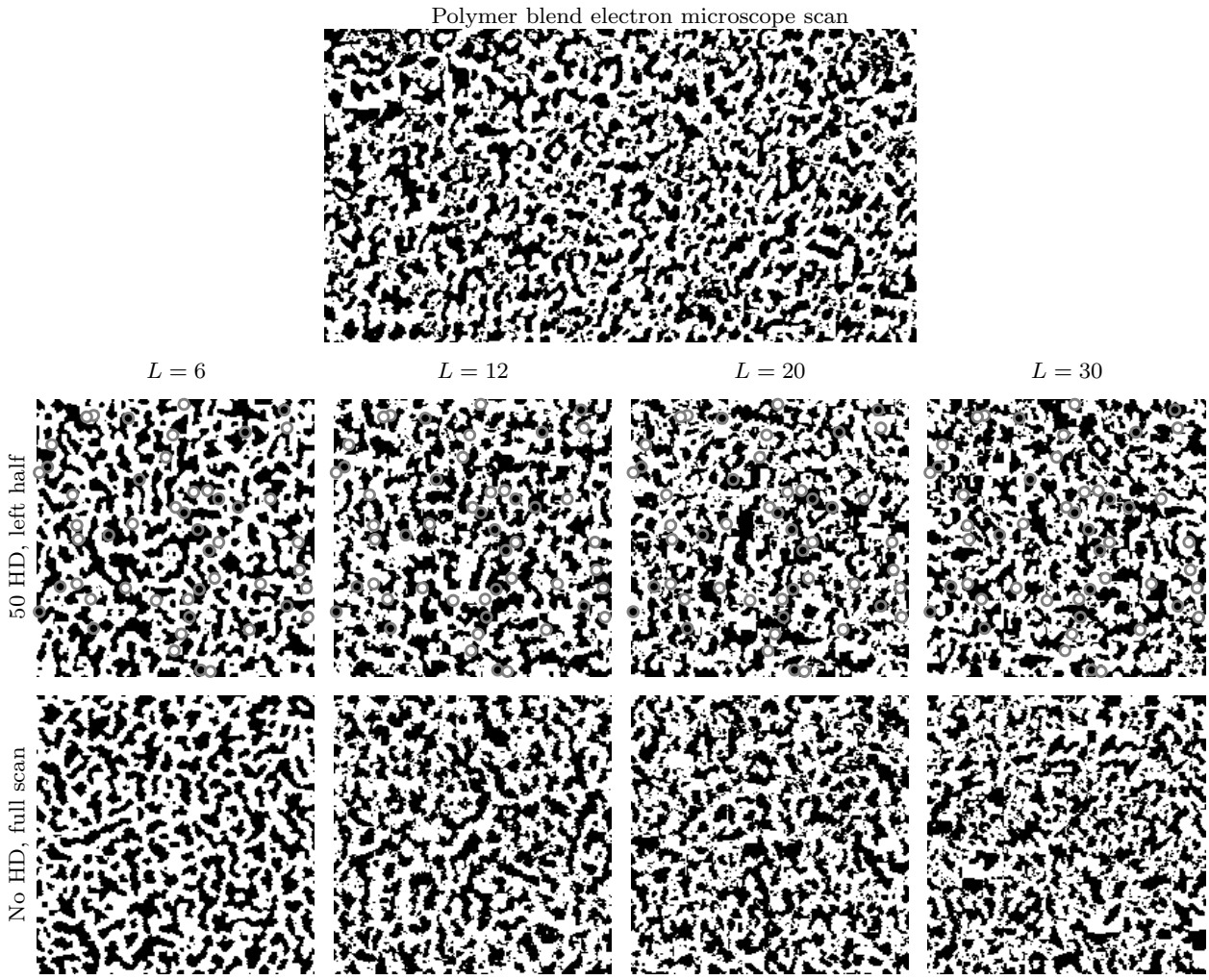


Figure 5.8 The reference image (top) was obtained by thresholding to black-and-white a scanning electron microscopy image of a polymer blend. In the first row, we display simulations based on the whole reference image, using no hard data. In the second row, we display simulations based on the reference image left-half, using the reference image right-half as a source for the 50 hard data. Each hard datum is indicated by a small gray annulus. The box size L used in the simulations varies from $L = 6$ (bottom row) to $L = 30$ (second row).

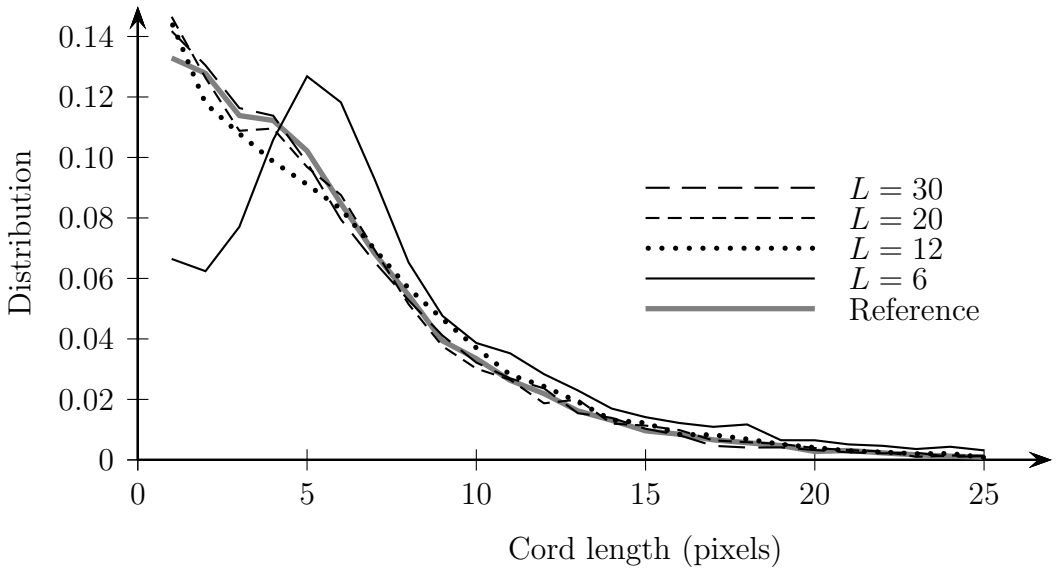


Figure 5.9 Cord length distributions of the unconstrained simulated images displayed in the first row of Figure 5.8.

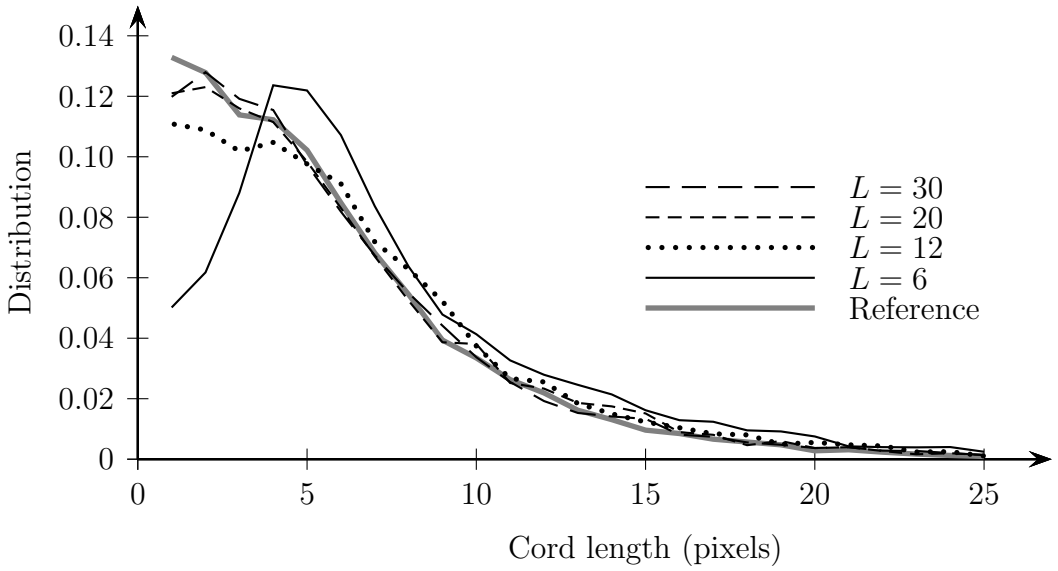


Figure 5.10 Cord length distributions of the constrained simulations displayed in the second row of Figure 5.8.

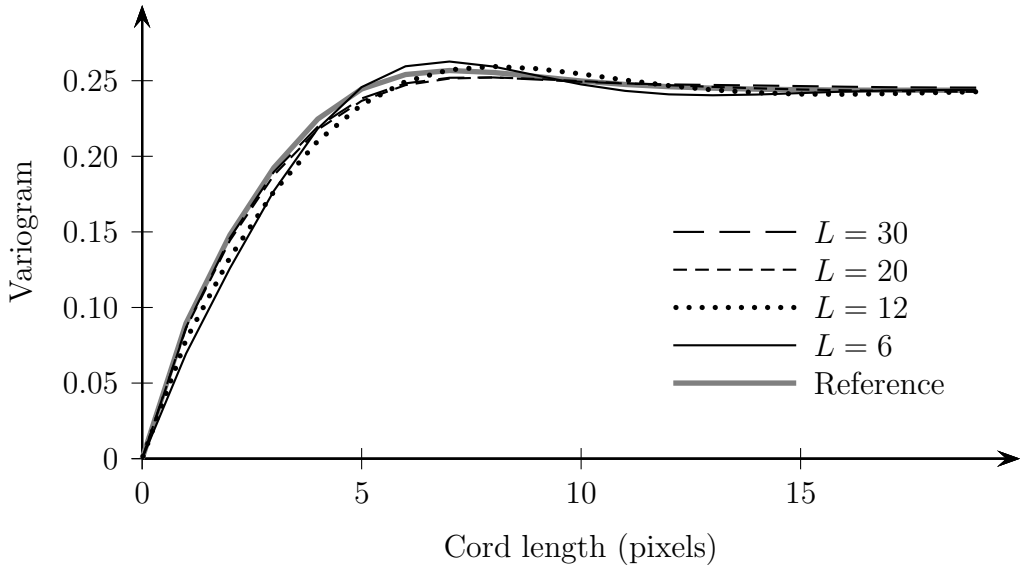


Figure 5.11 Variograms of the constrained simulations displayed in the second row of Figure 5.8.

For each box size $L \in \{6, 12, 20, 30\}$, the CLD and the variogram were obtained from a set of 10 independent simulations. The resulting standard deviation of each CLD relative frequency in the Figures 5.9 and 5.10 does not exceed 0.01.

Comparing the constrained and the unconstrained simulations (Figures 5.9 and 5.10), we observe that the constrained simulations exhibit fewer small cords (cord lengths of 1, 2 and 3 pixels) than the unconstrained simulations, i.e. 11-12% versus 13%. This effect is most significant for $L = 12$. For $L = 6$, we observe that both simulations suffer from a marked deficit of small cords and a excess of medium cords (about 5 pixels). This deficit of the number of small clusters is perceptible visually when comparing with the reference. A similar phenomenon was observed in the patchwork simulations (El Ouassini, Saucier, Marcotte and Favis, 2008).

Simulations using the larger box sizes $L = 20$ or 30 allow a good reproduction of the variogram, of the CLD and of the visual appearance of the original image patterns. However, unwanted discontinuities at the borders of assembled states become more visible for larger values of L. They take the form of square artifacts of size $L/2$ that resemble the artifacts observed in a lossy compressed jpeg image. These artifacts are a direct consequence of the poverty of the set $\mathcal{S}_r$ relatively to the set of all the possible states $\boldsymbol{s}$ of the underlying random

field. Indeed, as L increases, the number of possible states for a given random field increases exponentially whereas the reference image, which represents a sub-sample of the random field, stays constant in size.

Producing an image with a custom local-mean histogram

In this section, we test the ability of the CPMS to satisfy a target local-mean histogram that differs greatly from the reference image histogram. We set $L = 12$, $M = 8$ and the bin boundaries are chosen so that $p_r(i) \simeq 1/8$, $i = 1, \ldots, 8$ (Figure 5.12a). The first simulation (Figure 5.12b) aims at producing boxes that are either all-black or all-white, using the target probabilities $\boldsymbol{p}_t = (47\%, 1\%, 1\%, 1\%, 1\%, 1\%, 1\%, 47\%)$. The second simulation, displayed in Figure 5.12c, aims at producing a histogram having a peak around $\phi \simeq 0.6$, using $\boldsymbol{p}_t = (1\%, 1\%, 1\%, 47\%, 47\%, 1\%, 1\%, 1\%)$. Even though these two custom simulations fail to pass the chi-square test, we observe in the Figures 5.12b-c that the resulting histograms have a shape which in each case is close to the target histogram. These simulations show that significant textural differences can be obtained by setting different values for the target histogram $\boldsymbol{p}_t$. In particular, the cluster size distribution can be quite sensitive to $\boldsymbol{p}_t$.

5.8.5 Simulations of a channel network

Conditional simulations

In this section, we test the ability of the CPMS to reproduce long-range connected structures such as channel networks (Figures 5.13 and 5.14). We use two bins for the local-mean histogram control and we set $\boldsymbol{p}_t = \boldsymbol{p}_r$, i.e. we want the channel area density to be about the same as the density observed in the reference image. The latter, displayed on the top-left of the Figures 5.13 and 5.14, is composed of channels having a constant width of six pixels. This reference image is not isotropic and its correlation lengths are long compared to the images considered so far, i.e. about 40 pixels in the horizontal direction and 80 pixels in the vertical direction. We examine the simulations quality as a function of two parameters: the number of hard data and the box size L. We consider simulations with 0, 20 and 200 hard data and $L \in \{10, 20, 30\}$. The hard data are drawn randomly from the top-right image displayed

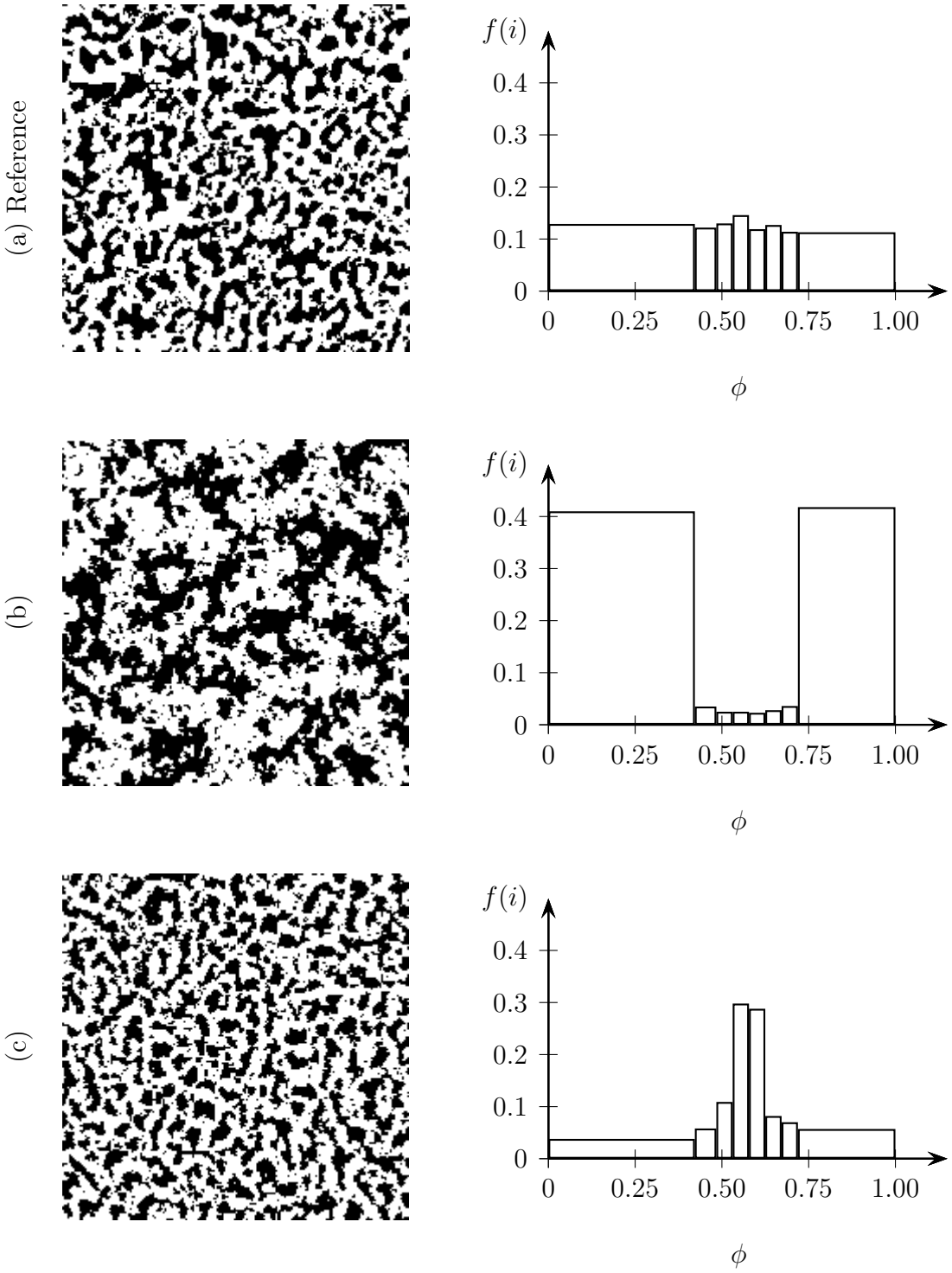


Figure 5.12 Simulations using $L = 12$ with their corresponding custom histograms. For each histogram, the height of bin i, which corresponds to the interval $[\varphi_{i-1}, \varphi_i]$, is $f(i)$.

in the Figures 5.13 and 5.14. Note that this hard data source image, while being visually similar to the reference image, can differ slightly from the reference image by its statistical properties (channel orientations, local-mean histogram, etc.). Single realizations using the CPMS are displayed in Figure 5.13. In Figure 5.14, we display ensemble averages termed E-type estimates (Deutsch and Journel, 1998). Each E-type estimate is the arithmetic mean of 25 realizations. E-type estimates can be used e.g. to estimate the channels locations for a given set of hard data.

In Figure 5.13, we observe that the channels reproduction quality improves as L increases. The first column of Figure 5.13 shows that using the relatively small box size $L = 10$ does not allow a good reproduction of the channel network. In contrast, we showed that the unilateral PSM (Faucher, Saucier and Marcotte, 2012) can prolong and reconnect channels correctly even with a small box size, i.e. $L = 12$. Using the larger box sizes $L = 20, 30$ (second and third column of Figure 5.13) allows to produce well connected channels and, as shown by the E-type estimates (Figure 5.14), also allows to extrapolate the channels locations. As previously, square artifacts of size $L/2$ are more visible for larger values of L.

The independence of different realizations of the CPMS can be tested via the standard deviation σ of the pixels values of an E-type estimate. Indeed, if the 25 realizations used to obtain one E-type estimate are independent, then $\sigma = \frac{\sigma_{\text{single}}}{\sqrt{25}}$, where σ_{single} is the standard deviation of the pixels values of a single realization. A single realization has pixel values in $\{0, 1\}$ and contain 40% and 60% of 0 and 1 respectively, which implies that $\sigma_{\text{single}} = 0.49$. Hence we expect $\sigma = \frac{0.49}{\sqrt{25}} \simeq 0.10$ for the E-type estimate. The estimated values of σ obtained for unconditional simulations, displayed in the first line of Table 5.1 (no hard-data), agree with this prediction.

To compare the E-type estimates with the hard data source image, we computed the correlation coefficient ρ between the pixel values in an E-type estimate and the corresponding pixel values in the hard data source image. In Table 5.1, we observe that ρ increases as the number of hard data increases, e.g. for $L = 20$ we obtained $\rho = 1.1\%$, 11%, and 49% with 0, 20 and 200 hard data respectively. This is to be expected since adding information by increasing the number of hard data should result in more accurate E-type estimates.

In Table 5.1, we observe that ρ also varies as a function of the box size L. This observation

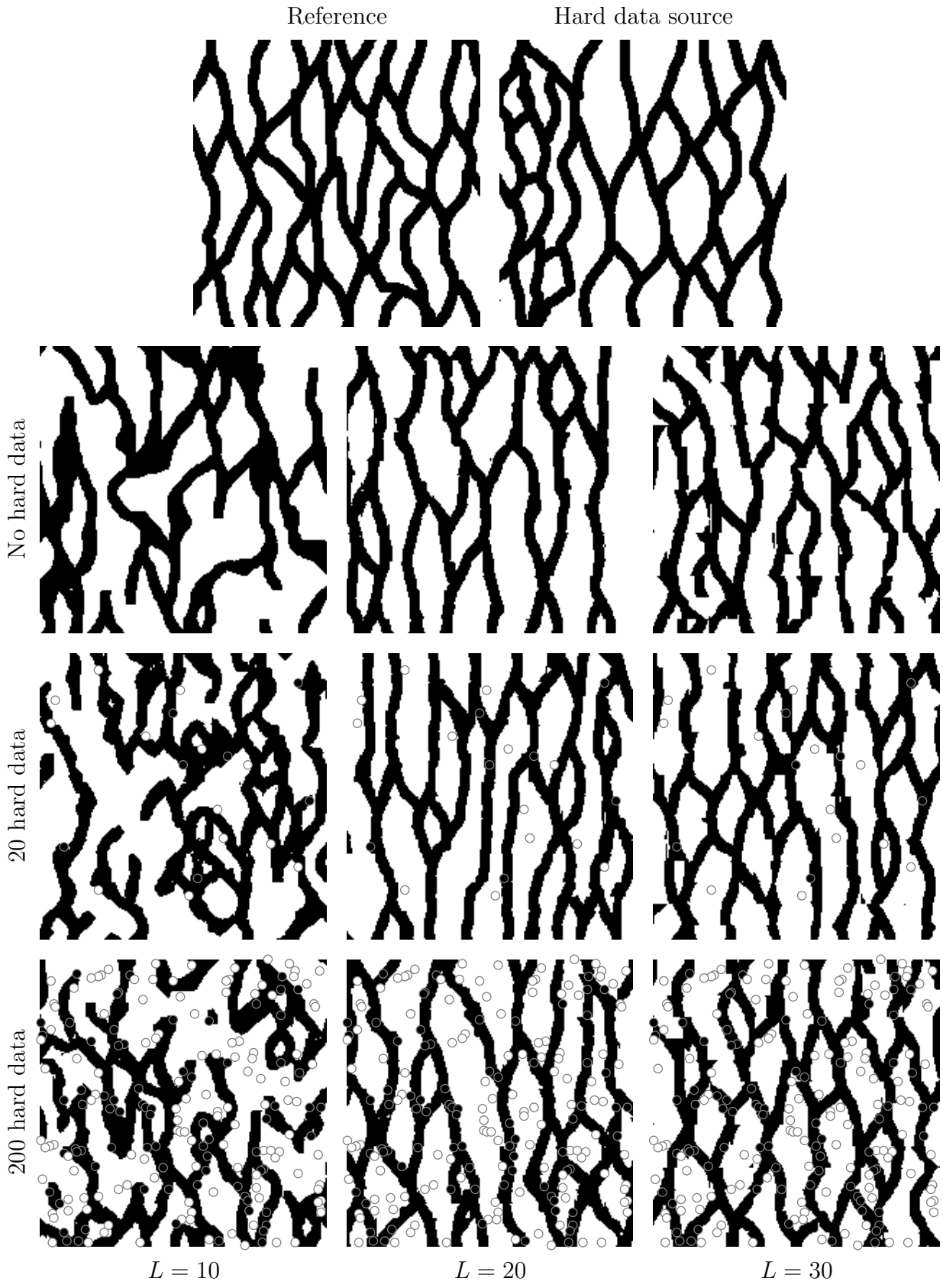


Figure 5.13 CPMS simulations based on the top-left reference image, which is composed of interconnected channel patterns.

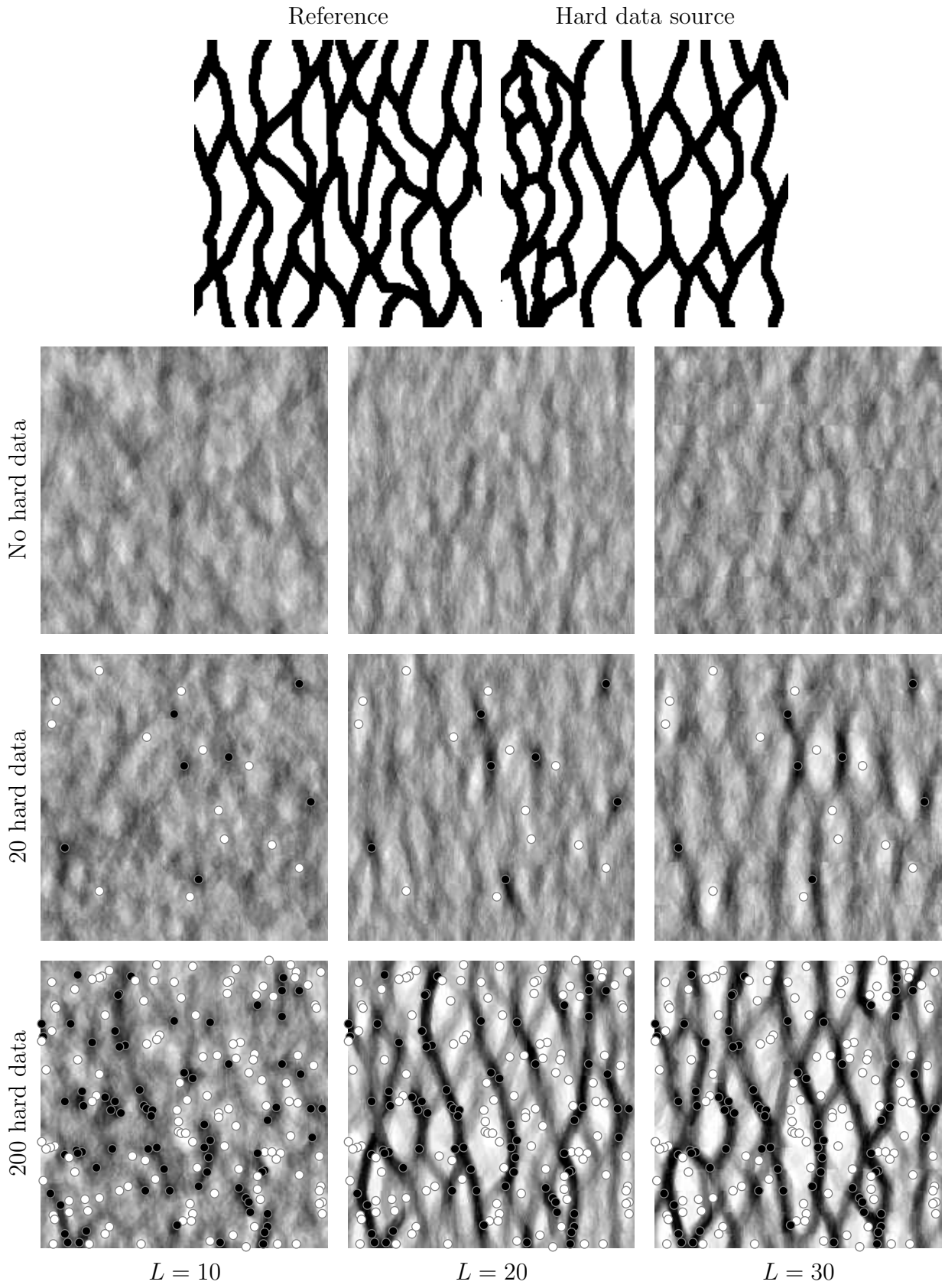


Figure 5.14 CPMS simulations based on the top-left reference image, which is composed of interconnected channel patterns: E-type estimates obtained with 25 independent simulations.

Table 5.1 Statistics of the E-type estimates (Figure 5.14). σ is the pixel value standard deviation and ρ is the pixel value correlation coefficient with the corresponding pixel in the hard data source image.

	$L = 10$		$L = 20$		$L = 30$	
	σ	ρ	σ	ρ	σ	ρ
No H.D.	0.10	0.7 %	0.10	1.1 %	0.10	0.6 %
20 H.D.	0.11	17 %	0.13	11 %	0.16	11 %
200 H.D.	0.18	36 %	0.27	49 %	0.30	45 %

can be used to select an appropriate box-size L for the simulations, i.e. a box size that maximizes the correlation coefficient between the E-type estimate and the hard-data source image. With 20 hard data, we observe in Table 5.1 that the E-type estimate obtained for $L = 10$ has a larger correlation coefficient than the E-type estimates obtained for $L = 20$ and 30 (17% versus 11% and 11% respectively). This higher correlation is obtained in spite of the fact that simulations obtained for $L = 20$ or $L = 30$ look more faithful to the hard data source image, as can be seen in Figure 5.13 (third row). With 200 hard data, we observe in Table 5.1 that the E-type estimate obtained for $L = 20$ has a larger correlation coefficient than the E-type estimates obtained with $L = 10$ and 30 (49% versus 36% and 45% respectively). For $L = 20$, it is observed in Figure 5.13 that simulations look faithful to the hard data source image. It is also observed in Figure 5.14 that the E-type estimate indicates channel locations fairly clearly.

These observations indicate that for a given reference image, there is an upper bound on the scale L beyond which the simulations accuracy decreases. This reduced accuracy is caused by the decreasing number of states available for patching in the set $\mathcal{S}_r$ as L increases. This poverty of $\mathcal{S}_r$ causes two effects: firstly, the simulations variability is reduced; secondly, simulations become *biased* in the sense that they are more likely to reproduce exactly, i.e. without any change, the large-scale patterns of the reference image. It is this bias that explains why the simulations with 20 hard data obtained for $L = 10$ are actually more accurate (but not as good-looking) than the simulations obtained for $L = 20$ or 30.

Comparison with the patchwork simulation method

In Figure 5.15, we present a comparison of images simulated with the CPMS and the patchwork simulation method (PSM) of Faucher, Saucier and Marcotte, (2012). All simulations are based on the reference image displayed on top of Figure 5.15, which is composed of channels oriented in the north-east diagonal direction. It was shown by Faucher, Saucier and Marcotte (2012) that this orientation helps reproducing correctly the channel network geometry when using the patchwork simulation method going from left to right and top to bottom. Both methods use two bins for the local-mean histogram control and $\boldsymbol{p}_t = \boldsymbol{p}_r$ for the target probabilities. The box sizes are $L = 16$ and $L = 12$ for the PSM and CPMS respectively, i.e. both methods construct the image by patching square-shaped pieces of size 12 pixels (see chapter 4 for the details of the construction method used in the PSM). Single realizations are displayed on the third row. The E-type estimates displayed in Figure 5.15 (second row) are the average of 256 independent realizations. The E-type estimates and the single realizations displayed in the first and second columns of Figure 5.15 are subject to respect hard-data that form a vertical line of white pixels. The E-type estimates and the single realizations displayed in the third and forth columns are subject to respect hard-data that form a vertical line of black pixels.

In Figure 5.15, we observe that the CPMS does not exhibit the main bias of the patchwork simulation method, which is an anisotropy that is most easily observed in the presence of conditioning hard data. For instance, in the simulations with the vertical white line of hard data, we observe that the patchwork simulation method is highly likely to create a black channel on the right-hand side of the vertical white line, whereas the CPMS rather creates one channel at each end of the vertical white line.

We also observe in Figure 5.15 that the CPMS method slightly reduces but does not eliminate the bias caused by the simulation grid. This bias, which takes the form of a grid-pattern observable in the E-type estimates, is particularly visible in the neighborhood of the conditioning data.

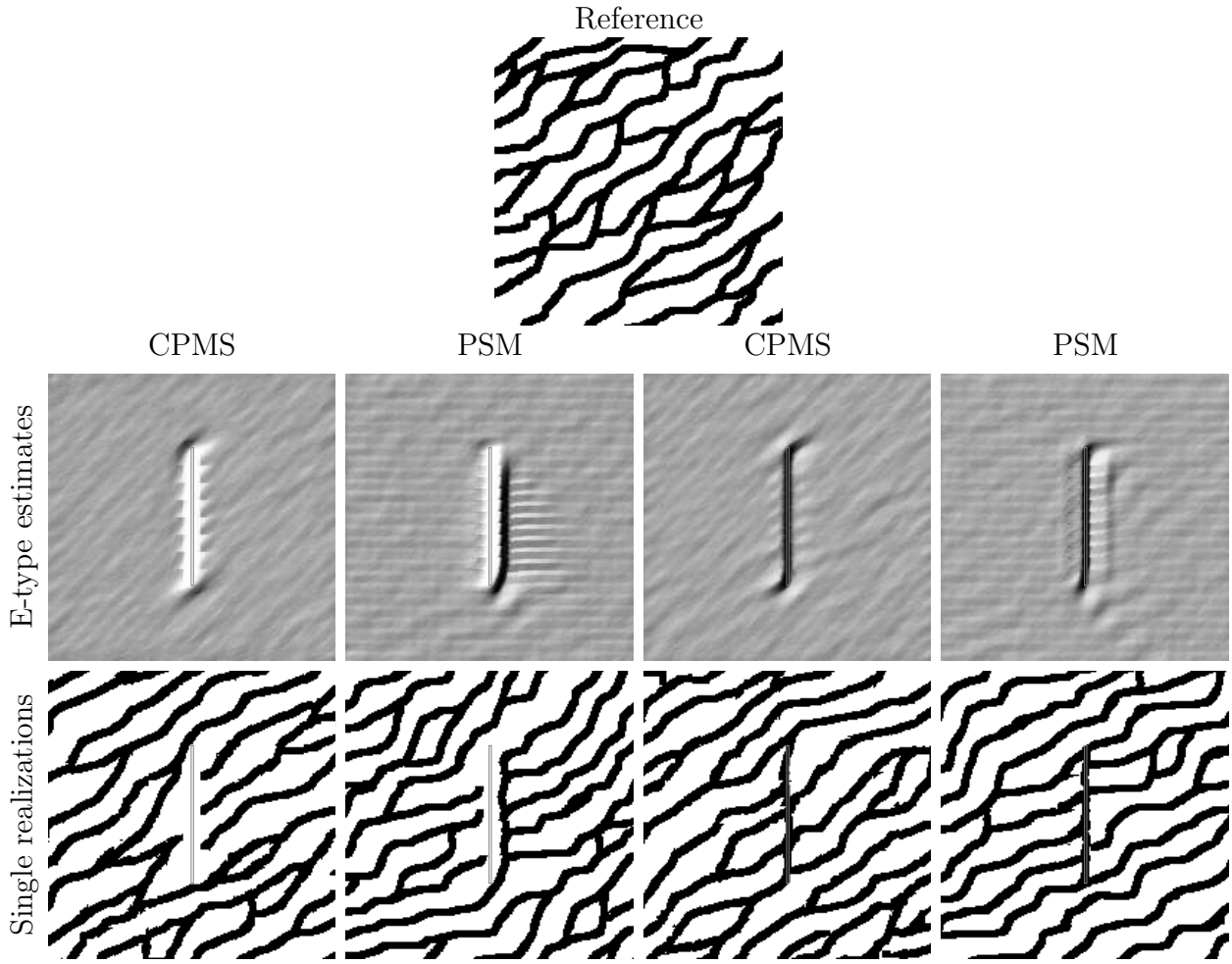


Figure 5.15 Comparison of simulations based on the top reference image, which is composed of interconnected channel patterns. The columns compare the CPMS and the unilateral PSM methods. The second and third rows display E-type estimates and single realizations respectively. In the first and second columns, the hard data take the form of a vertical white line. In the third and fourth columns, the hard data take the form of a vertical black line.

5.9 Concluding remarks

We proposed a simulation method call *corrective pattern-matching simulation* (CPMS) that is inspired from three main sources: the patchwork simulation method (PSM) introduced by El Ouassini, Saucier, Marcotte and Favis (2008); the patchwork simulation methods with control of the local-mean histogram introduced by Faucher, Saucier and Marcotte (2012); the conditional simulation with patterns introduced by Arpat and Caers (2007). It was shown by Faucher, Saucier and Marcotte (2012) that the approach used in the patchwork simulation method, i.e. a sequential construction that proceeds unilaterally, does not take into account hard data in the best possible way. Indeed, the image regions that are more strongly constrained by the presence of hard data should be constructed first. From this standpoint, the CPMS constructs the image in a non-unilateral manner: the method starts by constructing the image in the regions that contain non-respected hard data and then continues the image construction by suitably assembling square-shaped images pieces. Proceeding in a non-unilateral manner also allows to eliminate the anisotropy observed in the PSM.

We defined our simulations as solutions of an optimization problem for which the goal is to minimize pattern-conformity errors in the simulated image while respecting two constraints: honoring the hard data and respecting a target local-mean histogram. We chose to tackle this problem by defining an auxiliary optimization problem in which the constraints take the form of penalization functions added to the objective function to minimize.

Our simulations successfully reproduce the small scale patterns found in the reference image as long as the box size L is larger or equal to the average pattern diameter. We observed that our simulation produces images that honor the hard data and reproduce correctly the local-mean histogram. This control of the local-mean histogram also results in a controlled variogram for all scales smaller than the box size L. The conditional simulations of channel networks demonstrated the ability of the CPMS to extrapolate and reconnect correctly large-scale connected structures. We also demonstrated, via the simulations of a polymer blend image, the ability of the CPMS to customize the local-mean histogram and, thus, the general aspect of the simulated image.

Multiple-point simulations such as *snesim* can control one-point statistics such as the

proportions of single pixel facies (Strebelle, 2000; Tuanfeng, Stein and McCormick, 2008). Our control of the local-mean histogram, which captures the image variability in a region of size L, is a strength of our approach. Such a control cannot be obtained easily in the simulation methods for which pixels are simulated one at a time.

The CPMS accuracy is limited by the size of the reference image that provides the bank of patterns that are assembled together to produce a simulation. The reference image size can be increased at the cost of a larger memory requirement for the storage of the search trees. As the box size L increases, the number of possible states in the underlying random field increases exponentially whereas the size of the bank of patterns decreases. It follows that pattern-continuity errors can become more visible in the simulated image as L increases. If L is too large, then simulations can become biased in the sense that they are more likely to reproduce exactly, i.e. without any change, the large-scale patterns of the reference image. For a given set of hard-data, the values of L that maximize the correlation coefficient between the E-type estimate and a synthetic hard-data source image are suitable candidates for the box size L.

The CPMS is a non-unilateral simulation method based on iterative image corrections performed on regions having a fixed size L that is significantly smaller than the image size. It follows from these characteristics that the statistical properties of the images produced by the CPMS can be excellent at scales smaller than L (if the reference image is sufficiently large), but imperfect at scales larger than L. This limitation was illustrated by the simulations based on a checkerboard reference image, which converged to images composed of a few imperfectly stitched checkerboards.

Our simulation approach can be extended readily to three-dimensions because all concepts can be generalized directly, e.g. a square block in 2D becomes a cube in 3D. However, a 3D extension of this model might raise computational issues, e.g. memory requirements of the tree structure for states storage.

Acknowledgements: The authors thank B.D. Favis for providing the scanning electron microscopy image of a polymer blend. This work was supported by the NSERC Discovery research grants of the two coauthors.

5.A Evaluation of the conditional variances

$\sigma_i^2 = \mathrm{E}\left[\epsilon(i)^2 \middle| D^2 = R^2\right]$

The chi-square statistics (5.16) can be written as $D^2 = \sum_{i=1}^{M} X(i)^2$, where the centered and normalized random variables $X(i)$ are defined by

$$X(i) = \frac{n(i) - N p_t(i)}{\sqrt{N p_t(i)}},\ i = 1,\ ...,\ M. \tag{5.45}$$

Using (5.45), the relative frequency errors $\epsilon(i) = (n(i) - N\,p_t(i))/N$ can be expressed as a function of $X(i)$, i.e. $\epsilon(i) = \sqrt{p_t(i)/N}\ X(i)$. It follows that the σ_i^2 defined in (5.19) can be written in the equivalent form

$$\sigma_i^2 = \frac{p_t(i)}{N}\ \mathrm{E}\left[X(i)^2 \ \middle|\ \sum_{j=1}^{M} X(j)^2 = R^2\right]. \tag{5.46}$$

To evaluate (5.46), the joint probability distribution of the random variables $X(1), ..., X(M)$ must be determined.

In (5.45), the random variables $n(i)$ are distributed according to a multinomial probability law with parameters N and $p_t(i)$, $i = 1, ..., M$, hence their covariances are known to be given by $\mathrm{Cov}\,[n(i), n(j)] = -N\,p_t(i)\,p_t(j) + N\,p_t(i)\,\delta_{i,j}$, where $\delta_{i,j}$ denotes the Kronecker delta. It follows from (5.45) that the random variables $X(i)$ have zero mean and that their covariances are given by

$$\mathrm{Cov}\,[X(i), X(j)] = \delta_{ij} - \sqrt{p_t(i)\ p_t(j)} =: C_X(i,j). \tag{5.47}$$

Since the $X(i)$s are approximately gaussian for N sufficiently large (central limit theorem), the random vector $\boldsymbol{X} := (X(1), ..., X(M))^T$ can be approximated by a random vector $\boldsymbol{U}$ having a multivariate normal law, i.e.

$$\boldsymbol{X} \longrightarrow \boldsymbol{U} \sim N(\boldsymbol{0}, C_X) \tag{5.48}$$

as $N \to \infty$. In (5.48), the covariance matrix C_X, defined by (5.47), can be written in the

form

$$C_X := I_M - \boldsymbol{n}\,\boldsymbol{n}^T, \tag{5.49}$$

where I_M denotes the identity matrix of dimension $M \times M$ and $\boldsymbol{n} := (\sqrt{p_t(1)}, ..., \sqrt{p_t(M)})^T$ is a vector.

Using (5.45), it can be checked that the constraint $\sum_{i=1}^{M} n(i) = N$, which is satisfied by the multinomial probability law, implies that $\sum_{i=1}^{M} \sqrt{p_t(i)}\, X(i) = 0$, i.e. the vector $\boldsymbol{X} \in \mathbb{R}^M$ belongs to the plane $\mathcal{P}$ of dimension $M-1$ that contains the origin and that is normal to the vector $\boldsymbol{n}$. Let us denote by $\mathcal{B} := \{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_{M-1}\}$ an arbitrary orthonormal basis of $\mathcal{P}$. Any $\boldsymbol{x} \in \mathcal{P}$ can be expressed in the parametric form $\boldsymbol{x} = B\,\boldsymbol{y}$, where B is a rectangular matrix of dimension $M \times (M-1)$ composed of the vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_{M-1}$ arranged in columns, i.e. $B_{i,j} := b_j(i)$, $i = 1, ..., M$, $j = 1, ..., M-1$, and where the vector $\boldsymbol{y} = (y(1), ..., y(M-1))^T$ contains the coordinates of $\boldsymbol{x}$ in the basis $\mathcal{B}$. Note that the orthonormality of $\mathcal{B}$ implies that $B^T B = I_{M-1}$. We define the random vector $\boldsymbol{Y} := (Y(1), ..., Y(M-1))^T$, where the Gaussian random variables $Y(i) \sim N(0,1)$, $i = 1, ..., M-1$ are mutually independent. In the following, it will be shown that

$$\boldsymbol{Z} := B\,\boldsymbol{Y} \sim N(\boldsymbol{0}, C_X), \tag{5.50}$$

and therefore the random vectors $\boldsymbol{Z}$ and $\boldsymbol{U}$ have the same multivariate probability distribution, i.e. $\boldsymbol{Z} \overset{d}{=} \boldsymbol{U}$.

The result (5.50) can be derived as follows. Since a linear combination of independent Gaussian random variables is Gaussian, the parametrization (5.50) implies that each $Z(i)$ is a Gaussian random variable with zero mean. If we introduce the orthonormal square matrix Σ composed of the vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_{M-1}$ and $\boldsymbol{n}$ arranged in columns, i.e. Σ is the matrix B augmented with the normal vector $\boldsymbol{n}$, then $\Sigma\Sigma^T = I_M$ and the covariance matrix C_Z of $\boldsymbol{Z}$

can be derived as follows:

$$\begin{aligned} C_Z &= \mathrm{E}\left[\boldsymbol{Z}\boldsymbol{Z}^T\right] \\ &= \mathrm{E}\left[B\ \boldsymbol{Y}\boldsymbol{Y}^T\ B^T\right] \\ &= B\ \mathrm{E}\left[\boldsymbol{Y}\boldsymbol{Y}^T\right]\ B^T \\ &= B\, I_{M-1} B^T \\ &= B\ B^T \\ &= \underbrace{B\ B^T + \boldsymbol{n}\,\boldsymbol{n}^T}_{=\Sigma\Sigma^T} - \boldsymbol{n}\,\boldsymbol{n}^T \\ &= I_M - \boldsymbol{n}\,\boldsymbol{n}^T, \end{aligned} \tag{5.51}$$

which is identical to C_X according to (5.49), and therefore $\boldsymbol{Z}$ and $\boldsymbol{U}$ have the same probability law.

The expected value in (5.46) can be written in the equivalent form $\mathrm{E}\left[X(i)^2 \;\middle|\; \boldsymbol{X}^T\boldsymbol{X} = R^2\right]$. The random vector $\boldsymbol{X}$ can be approximated by $\boldsymbol{U}$ as $N \to \infty$ (according to (5.48)) and $\boldsymbol{U} \overset{d}{=} \boldsymbol{Z}$, hence

$$\begin{aligned} \mathrm{E}\left[X(i)^2 \;\middle|\; \boldsymbol{X}^T\boldsymbol{X} = R^2\right] &\approx \mathrm{E}\left[U(i)^2 \;\middle|\; \boldsymbol{U}^T\boldsymbol{U} = R^2\right] \\ &= \mathrm{E}\left[Z(i)^2 \;\middle|\; \boldsymbol{Z}^T\boldsymbol{Z} = R^2\right]. \end{aligned} \tag{5.52}$$

On the one hand, it follows from (5.50) that $\boldsymbol{Z}^T\boldsymbol{Z} = \boldsymbol{Y}^T B^T B\, \boldsymbol{Y} = \boldsymbol{Y}^T\boldsymbol{Y}$ because $B^T B = I_{M-1}$. On the other hand, since $B_{i,j} = b_j(i)$, the definition (5.50) implies that $Z(i) = \sum_{j=1}^{M-1} b_j(i)\, Y(j)$ for each i. It follows that (5.52) takes the form

$$\begin{aligned} \mathrm{E}\left[X(i)^2 \;\middle|\; \boldsymbol{X}^T\boldsymbol{X} = R^2\right] &\approx \mathrm{E}\left[\left(\sum_{j=1}^{M-1} b_j(i)\, Y(j)\right)^2 \;\middle|\; \boldsymbol{Y}^T\boldsymbol{Y} = R^2\right] \\ &= \sum_{m=1}^{M-1}\sum_{n=1}^{M-1} b_m(i)\, b_n(i)\ \mathrm{E}\left[Y(m)\, Y(n) \;\middle|\; \boldsymbol{Y}^T\boldsymbol{Y} = R^2\right]. \end{aligned} \tag{5.53}$$

The independence of the $Y(j)$s implies that the probability density function (PDF) of the random vector $\boldsymbol{Y}$ is of the form $f(\boldsymbol{y}) = e^{-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}}/(2\pi)^{(M-1)/2}$, hence f is constant on the surface of the sphere of equation $\boldsymbol{y}^T\boldsymbol{y} = R^2$. Moreover, the conditional PDF required to compute the expected values in (5.53) is also constant on this sphere. If $n \neq m$, then this

symmetry implies that

$$\mathrm{E}\left[Y(m)\,Y(n)\;\middle|\;\boldsymbol{Y}^T\boldsymbol{Y}=R^2\right]=0,\ n\neq m. \tag{5.54}$$

If $n=m$, then the same symmetry implies that $\mathrm{E}\left[Y^2(n)\;\middle|\;\boldsymbol{Y}^T\boldsymbol{Y}=R^2\right]$ is independent of n. It follows that

$$\begin{aligned}\mathrm{E}\left[Y^2(n)\;\middle|\;\boldsymbol{Y}^T\boldsymbol{Y}=R^2\right] &= \frac{1}{M-1}\sum_{n=1}^{M-1}\mathrm{E}\left[Y^2(n)\;\middle|\;\boldsymbol{Y}^T\boldsymbol{Y}=R^2\right]\\ &= \frac{1}{M-1}\,\mathrm{E}\left[\sum_{n=1}^{M-1}Y^2(n)\;\middle|\;\boldsymbol{Y}^T\boldsymbol{Y}=R^2\right]\\ &= \frac{1}{M-1}\,\mathrm{E}\left[\boldsymbol{Y}^T\boldsymbol{Y}\;\middle|\;\boldsymbol{Y}^T\boldsymbol{Y}=R^2\right]=\frac{R^2}{M-1}.\end{aligned} \tag{5.55}$$

Substituting (5.54) and (5.55) into (5.53) yields

$$\mathrm{E}\left[X(i)^2\;\middle|\;\boldsymbol{X}^T\boldsymbol{X}=R^2\right]\;\approx\;\frac{R^2}{M-1}\sum_{n=1}^{M-1}\left(b_n(i)\right)^2. \tag{5.56}$$

Denoting the canonical basis of $\mathbb{R}^M$ by $\{\boldsymbol{e}_1,\boldsymbol{e}_2,\ldots,\boldsymbol{e}_M\}$, we observe that

$$\begin{aligned}\sum_{n=1}^{M-1}\left(b_n(i)\right)^2 &= \sum_{n=1}^{M-1}\left(\boldsymbol{b}_n\cdot\boldsymbol{e}_i\right)^2\\ &= \underbrace{\sum_{n=1}^{M-1}\left(\boldsymbol{b}_n\cdot\boldsymbol{e}_i\right)^2+\left(\boldsymbol{n}\cdot\boldsymbol{e}_i\right)^2}_{=\|\boldsymbol{e}_i\|^2}-\left(\boldsymbol{n}\cdot\boldsymbol{e}_i\right)^2\\ &= 1-p_t(i),\end{aligned} \tag{5.57}$$

where we used, in the second line of (5.57), the fact that $\mathcal{B}\cup\{\boldsymbol{n}\}$ is an orthonormal basis of $\mathbb{R}^M$. Using (5.56) and (5.57), (5.46) takes the final form

$$\sigma_i^2\approx\frac{p_t(i)(1-p_t(i))}{N}\;\frac{R^2}{M-1}. \tag{5.58}$$

5.B Existence of a pair of indices (i,j) such that $\epsilon(i)-\epsilon(j)>\sigma_\epsilon$ if $D^2>R^2$

We assume that $D^2>R^2$, which is possible, and we make the hypothesis that it is simultaneously possible to have $\epsilon^2(i)\leq\sigma_i^2$ for all $i\in I$, where $I:=\{1,\ldots,M\}$. Using

$D^2 = N \sum_{i=1}^{M} \frac{\epsilon^2(i)}{p_t(i)}$, we observe that

$$\epsilon^2(i) \leq \sigma_i^2 \text{ for all } i \in I \quad \Rightarrow \quad D^2 \leq N \sum_{i=1}^{M} \frac{\sigma_i^2}{p_t(i)} = R^2, \tag{5.59}$$

where the equality was obtained by substituting σ_i^2 by the expression (5.19). This result, i.e. $D^2 \leq R^2$, contradicts the assumption $D^2 > R^2$ and therefore the hypothesis $\epsilon^2(i) \leq \sigma_i^2$ for all $i \in I$ is false. We conclude that if $D^2 > R^2$, then there is at least one histogram bin of index $k \in I$ such that $\epsilon^2(k) > \sigma_k^2 \Leftrightarrow |\epsilon(k)| > \sigma_k$.

As a consequence, there always exists at least one pair of indices $(i, j) \in I \times I$ such that $\epsilon(i) - \epsilon(j) > \sigma_\epsilon$, where $\sigma_\epsilon := \min_{i \in \{1,...,M\}} \sigma_i$. This result can be proved as follows. Since $\sum_{i=1}^{M} \epsilon(i) = 0$, there exists at least one pair of indices $(m, n) \in I \times I$ such that $\epsilon(m) < 0$ and $\epsilon(n) > 0$. If $\epsilon(k) > 0$, then

$$\epsilon(k) - \epsilon(m) > \epsilon(k) = |\epsilon(k)| > \sigma_k \geq \sigma_\epsilon,$$

hence the pair $(i, j) = (k, m)$ satisfies $\epsilon(i) - \epsilon(j) > \sigma_\epsilon$. If $\epsilon(k) < 0$, then

$$-\epsilon(k) + \epsilon(n) > -\epsilon(k) = |\epsilon(k)| > \sigma_k \geq \sigma_\epsilon,$$

hence the pair $(i, j) = (n, k)$ satisfies $\epsilon(i) - \epsilon(j) > \sigma_\epsilon$.

5.C Derivation of the expectation $\mathrm{E}\left[e_{\mathbf{stat}}(j) \middle| D^2 = R^2\right]$

The statistical error $e_{\mathrm{stat}}(j)$ is defined by

$$e_{\mathrm{stat}}(j) = \begin{cases} \epsilon(i(\boldsymbol{s}_j)) & \text{if } \epsilon(i(\boldsymbol{s}_j)) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.60}$$

The set of events $\{B_i := \{i(\boldsymbol{s}_j) = i\},\ i = 1,\ ...,\ M\}$ forms a partition for the random variable $i(\boldsymbol{s}_j)$, i.e. $i(\boldsymbol{s}_j) \in \bigcup_{i=1}^{M} B_i$ with $B_i \cap B_j = \emptyset$ for $i \neq j$. In the following, the event $D^2 = R^2$ will be denoted by A. Applying the law of total probability with the partition B_1, ..., B_M to

the evaluation of the expectation $\mathrm{E}\left[e_{\mathrm{stat}}(j)\middle| D^2 = R^2\right]$, we obtain successively

$$\begin{aligned}
\mathrm{E}\left[e_{\mathrm{stat}}(j)\middle| A\right] &= \sum_{i=1}^{M} \mathrm{E}\left[e_{\mathrm{stat}}(j)\,\middle|\, A,\, B_i\right] \mathrm{P}\left[B_i|A\right] \\
&= \sum_{i=1}^{M} \Bigg(\mathrm{E}\left[e_{\mathrm{stat}}(j)\,\middle|\, A,\, B_i,\, \epsilon(i) \geq 0\right]\ \mathrm{P}\left[\epsilon(i) \geq 0\middle| A\right] \\
&\quad + \underbrace{\mathrm{E}\left[e_{\mathrm{stat}}(j)\,\middle|\, A,\, B_i,\, \epsilon(i) < 0\right]}_{=0}\ \mathrm{P}\left[\epsilon(i) < 0\middle| A\right] \Bigg)\ \mathrm{P}\left[B_i|A\right] \\
&= \sum_{i=1}^{M} \mathrm{E}\left[\epsilon(i)\,\middle|\, A,\, \epsilon(i) \geq 0\right]\ \mathrm{P}\left[\epsilon(i) \geq 0\middle| A\right]\ \mathrm{P}\left[i(\boldsymbol{s}_j) = i\middle| A\right].
\end{aligned} \tag{5.61}$$

To evaluate the expected value and the two probabilities in the last line of (5.61), we will use the random vectors $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z} = B\,\boldsymbol{Y}$, as defined in the Appendix 5.A. Using these random vectors, the probability $\mathrm{P}\left[i(\boldsymbol{s}_j) = i\,\middle|\, A\right]$ in the last line of (5.61) can be evaluated as follows:

$$\begin{aligned}
\mathrm{P}\left[i(\boldsymbol{s}_j) = i\,\middle|\, A\right] &= \mathrm{E}\left[f(i)\middle| A\right] \\
&= \mathrm{E}\left[p_t(i) + \epsilon(i)\,\middle|\, D^2 = R^2\right] \\
&= p_t(i) + \mathrm{E}\left[\sqrt{\frac{p_t(i)}{N}}\ X(i)\,\middle|\, \boldsymbol{X}^T\boldsymbol{X} = R^2\right] \\
&\simeq p_t(i) + \mathrm{E}\left[\sqrt{\frac{p_t(i)}{N}}\ Z(i)\,\middle|\, \boldsymbol{Z}^T\boldsymbol{Z} = R^2\right] \\
&= p_t(i) + \sqrt{\frac{p_t(i)}{N}}\ \mathrm{E}\left[\sum_{j=1}^{M-1} Y(j)\ \boldsymbol{b}_j^T\boldsymbol{e}_i\,\middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2\right] \\
&= p_t(i) + \sqrt{\frac{p_t(i)}{N}}\ \sum_{j=1}^{M-1} \boldsymbol{b}_j^T\boldsymbol{e}_i\ \underbrace{\mathrm{E}\left[Y(j)\ \middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2\right]}_{=0} \\
&= p_t(i).
\end{aligned} \tag{5.62}$$

Note that in the sixth line of (5.62), each expectation vanishes because of the symmetry of the conditional PDF, which is constant on a sphere of radius R.

The probability $\mathrm{P}\left[\epsilon(i) \geq 0 \,\middle|\, A\right]$ in the last line of (5.61) can be evaluated as follows:

$$\begin{aligned} \mathrm{P}\left[\epsilon(i) \geq 0 \,\middle|\, A\right] &= \mathrm{P}\left[X(i) \geq 0 \,\middle|\, \boldsymbol{X}^T\boldsymbol{X} = R^2\right] \\ &\simeq \mathrm{P}\left[Z(i) \geq 0 \,\middle|\, \boldsymbol{Z}^T\boldsymbol{Z} = R^2\right] \\ &= \mathrm{P}\left[\sum_{j=1}^{M-1} Y(j)\ \boldsymbol{b}_j \cdot \boldsymbol{e}_i \geq 0 \,\middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2\right]. \end{aligned} \tag{5.63}$$

The summation in the last line of (5.63) can be simplified as follows. For any fixed $i \in \{1, \ldots, M\}$, we choose an orthonormal basis $\{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_{M-1}\}$ of the plane $\mathcal{P}$ (defined in appendix 5.A) such that the first vector $\boldsymbol{b}_1$ is the orthogonal projection of $\boldsymbol{e}_i$ on $\mathcal{P}$, i.e. $\boldsymbol{b}_1 := \frac{\mathrm{proj}_{\mathcal{P}}(\boldsymbol{e}_i)}{\|\mathrm{proj}_{\mathcal{P}}(\boldsymbol{e}_i)\|}$, and we denote the resulting basis by $\mathcal{B}_i$. In the orthonormal basis $\mathcal{B}_i \cup \{\boldsymbol{n}\}$, where $\boldsymbol{n}$ is the vector normal to the plane $\mathcal{P}$, the only non-zero components of the vector $\boldsymbol{e}_i$ are associated to the basis vectors $\boldsymbol{b}_1$ and $\boldsymbol{n}$, i.e. $\boldsymbol{b}_j^T\boldsymbol{e}_i = 0$ for all $j \neq 1$. It follows that (5.63) simplifies to

$$\begin{aligned} \mathrm{P}\left[\epsilon(i) \geq 0 \,\middle|\, A\right] &\simeq \mathrm{P}\left[Y(1)\ \boldsymbol{b}_1^T\boldsymbol{e}_i \geq 0 \,\middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2\right] \\ &= \mathrm{P}\left[Y(1)\ \geq 0 \,\middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2\right] \\ &= 1/2, \end{aligned} \tag{5.64}$$

where the last line of (5.64) is also a consequence of the symmetry of the conditional PDF, which is constant on a sphere of radius R.

To evaluate the expected value in the last line of (5.61) for a given i, we use again the basis $\mathcal{B}_i$, which leads to

$$\begin{aligned} \mathrm{E}\left[\epsilon(i) \,\middle|\, A,\, \epsilon(i) \geq 0\right] &= \sqrt{\frac{p_t(i)}{N}}\ \mathrm{E}\left[X(i) \,\middle|\, \boldsymbol{X}^T\boldsymbol{X} = R^2,\, X(i) \geq 0\right] \\ &\simeq \sqrt{\frac{p_t(i)}{N}}\ \mathrm{E}\left[Z(i) \,\middle|\, \boldsymbol{Z}^T\boldsymbol{Z} = R^2,\, Z(i) \geq 0\right] \\ &= \sqrt{\frac{p_t(i)}{N}}\ \boldsymbol{b}_1^T\boldsymbol{e}_i\ \mathrm{E}\left[Y(1) \,\middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2,\, Y(1) \geq 0\right], \end{aligned} \tag{5.65}$$

where

$$\boldsymbol{b}_1^T\boldsymbol{e}_i = \|\mathrm{proj}_{\mathcal{P}}(\boldsymbol{e}_i)\| = \|\boldsymbol{e}_i - (\boldsymbol{n}^T\boldsymbol{e}_i)\,\boldsymbol{n}\| = \sqrt{1 - p_t(i)}. \tag{5.66}$$

In the third line of (5.65), the condition $Y(1) \geq 0$ implies that the conditional PDF used for

the expected value of Y_1 is now distributed uniformly over the surface of a hemisphere H of radius R, where H is defined by $H = \{(y(1), ..., y(M-1)) \in \mathbb{R}^{M-1} : \sum_{j=1}^{M-1}(y(j))^2 = R^2, y(1) > 0\}$. This expected value takes the explicit form

$$\mathrm{E}\left[Y(1) \,\middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2,\, Y(1) \geq 0\right] = \int_H y(1)\,\rho\; dh, \tag{5.67}$$

where dh is an infinitesimal surface area element of the surface H and ρ denotes the constant conditional PDF. The normalization constraint on ρ implies that

$$\rho = \frac{1}{\int_H dh} = \frac{2}{S_{M-1}(R)}, \tag{5.68}$$

where $S_N(R)$ denotes the surface area of a N-sphere (i.e. the area of the surface of equation $\sum_{i=1}^{N} y(i)^2 = R^2$). The hemisphere H can be parametrized with the transformation

$$\begin{cases} y(n) &= y_n, \quad n \in \{1, ..., M-2\} \\ y(M-1) &= \sqrt{R^2 - \sum_{j=1}^{M-2} y_j^2}, \end{cases} \tag{5.69}$$

where the parameters $y_1, ..., y_{M-2}$ belong to the sphere $\Sigma := \{(y_1, ..., y_{M-2}) \in \mathbb{R}^{M-2} : \sum_{j=1}^{M-2} y_j^2 \leq R^2\}$. Using the parametrization (5.69), the volume element in (5.67) takes the form $dh = |J|\, dy_1 \ldots dy_{M-2}$, where the factor $|J|$ can be shown to be given by

$$|J| \;=\; \frac{R}{\sqrt{R^2 - \sum_{j=1}^{M-2} y_j^2}}. \tag{5.70}$$

For instance, if $M - 1 = 3$, then we have

$$\begin{cases} y(1) &= y_1 \\ y(2) &= y_2, \\ y(3) &= \sqrt{R^2 - y_1^2 - y_2^2} \end{cases} \tag{5.71}$$

where $y_1^2 + y_2^2 \leq R^2$. Defining $\boldsymbol{r}(y_1, y_2) := (y(1), y(2), y(3))$, where $y(1)$, $y(2)$ and $y(3)$ are

given by (5.71), the factor $|J|$ is equal to the norm of the fundamental vector product

$$|J| = \left\| \frac{\partial \boldsymbol{r}}{\partial y_1} \times \frac{\partial \boldsymbol{r}}{\partial y_2} \right\| = \left\| \begin{vmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 \\ 1 & 0 & \frac{-y_1}{\sqrt{R^2-y_1^2-y_2^2}} \\ 0 & 1 & \frac{-y_2}{\sqrt{R^2-y_1^2-y_2^2}} \end{vmatrix} \right\| = \frac{R}{\sqrt{R^2-y_1^2-y_2^2}}. \tag{5.72}$$

Using the transformation (5.69) and the value of ρ given by (5.68), the integral (5.67) takes the form

$$\begin{aligned} \mathrm{E}\left[Y(1) \,\middle|\, \boldsymbol{Y}^T\boldsymbol{Y} = R^2,\, Y(1) \geq 0\right] &= \rho \int_\Sigma y(1)\, |J|\, dy_2 \ldots dy_{M-1} \\ &= \rho\, R \int_\Sigma dy_2 \ldots dy_{M-1} \\ &= 2\, R \frac{V_{M-2}(R)}{S_{M-1}(R)} \\ &= \frac{R}{\sqrt{\pi}} \frac{\Gamma\left(\frac{M-1}{2}\right)}{\Gamma\left(\frac{M}{2}\right)}, \end{aligned} \tag{5.73}$$

where we used the fact that the volume and surface area of a N-sphere of radius R are given by $V_N(R) = \frac{2\,\pi^{N/2}}{N\ \Gamma(N/2)} R^N$ and $S_N(R) = \frac{2\,\pi^{N/2}}{\Gamma(N/2)} R^{N-1}$ respectively. Substituting the last line of (5.73) and (5.66) into (5.65) yields

$$\begin{aligned} \mathrm{E}\left[\epsilon(i) \middle| A,\, \epsilon(i) > 0\right] &= \sqrt{\frac{p_t(i)\big(1-p_t(i)\big)}{N}}\ \mathrm{E}\left[Y(1) \,\middle|\, A,\, Y(1) \geq 0\right] \\ &= \underbrace{\sqrt{\frac{p_t(i)\big(1-p_t(i)\big)}{N} \frac{R^2}{M-1}}}_{\sigma_i}\ \underbrace{\sqrt{\frac{M-1}{\pi}} \frac{\Gamma\left(\frac{M-1}{2}\right)}{\Gamma\left(\frac{M}{2}\right)}}_{\beta(M)} \\ &= \sigma_i\ \beta(M), \end{aligned} \tag{5.74}$$

where σ_i is given by (5.19) and $\beta(M)$ is defined by

$$\beta(M) := \sqrt{\frac{M-1}{\pi}}\ \frac{\Gamma\left(\frac{M-1}{2}\right)}{\Gamma\left(\frac{M}{2}\right)}. \tag{5.75}$$

Note that for $M \geq 2$ the coefficient $\beta(M)$ satisfies $\sqrt{2/\pi} \leq \beta(M) \leq 1$. Finally, Substituting

the expected value (5.74) and the probabilities (5.62) and (5.64) into (5.61), we obtain

$$\mathrm{E}\left[e_{\mathrm{stat}}(j)\middle|D^2 = R^2\right] \;=\; \frac{\beta(M)}{2}\sum_{i=1}^{M}\sigma_i\, p_t(i). \tag{5.76}$$

CHAPITRE 6

APPROCHE MULTI-ÉCHELLE DE SIMULATION PAR RAPIÉÇAGE

À la section 5.8.5, nous expliquons qu'utiliser une boîte de taille fixe L ne permet pas de produire des chenaux lisses tout en reproduisant la structure globale des chenaux. Plus précisément, nous ne pouvons pas reproduire la structure à grande échelle et la structure à petite échelle. Pour être en mesure de reproduire les structures sur une plus grande gamme d'échelles, nous employons une approche multi-échelle. Plus précisément, nous faisons des simulations successives de la même image en employant différentes tailles de boîtes L_i, $i = 0, 1, \ldots$ Nous commençons avec une taille L_0 suffisamment grande pour reproduire la structure globale de l'image. Ensuite, partant de l'image résultant de cette première simulation, nous choisissons une taille L_1 plus petite et nous réappliquons notre méthode de simulation. Finalement, nous répétons le processus tant que l'image présente des irrégularités dans sa structure fine. De plus, afin d'éviter que la simulation des échelles inférieures ne détruise la structure des échelles supérieures, nous fixons la valeur de certains points de l'image après chaque passage de la simulation, c'est-à-dire que certaines données seront considérées comme des données conditionnantes (hard data) pour les simulations aux échelles inférieures.

Notre approche multi-échelle diffère donc substantiellement des autres simulations multi-échelles présentées au chapitre 1. Par exemple, la simulation point par point *snesim* de Strebelle commence par simuler les pixels situés sur la première grille (un sous-ensemble de pixels distants les uns des autres). Ensuite, les pixels de la deuxième grille sont simulés conditionnellement aux pixels déjà simulés sur la première grille. La simulation *snesim* continue ainsi jusqu'à ce que toutes les grilles aient été simulées. Dans la méthode *snesim*, les grilles forment une partition de l'ensemble des pixels de l'image et chaque pixel n'est simulé qu'une seule fois.

Dans la méthode *simpat* d'Arpat et Caers, bien que tout le motif à grande échelle soit collé à l'image (carré $L_0 \times L_0$), il n'y a qu'un sous-ensemble des pixels qui est utilisé pour le calcul de la distance (voir section 1.6 et figure 1.6). De plus, dans la méthode *simpat*, l'image

résultante d'une échelle supérieure est utilisée telle quelle sans conditionnement dur (hard data) pour la simulation de l'échelle suivante.

6.1 Choix de l'échelle de base

La première étape, pour une simulation multi-échelle, est de déterminer une taille appropriée L_0 pour la simulation de la plus grande échelle. Cette taille L_0 doit être suffisamment grande pour reproduire la structure globale observée dans l'image de référence. Cette valeur de L_0 est typiquement de l'ordre de grandeur de la longueur de corrélation. En pratique, nous basons le choix de la taille L_0 sur deux considérations :

1. choisir L_0 trop petit nous empêche de reproduire la structure globale du milieu observé,
2. choisir L_0 trop grand entraîne une raréfaction des états disponibles et, par conséquent, une faible variabilité entre les simulations, c'est-à-dire un faible niveau d'indépendance entre des simulations consécutives.

Dans le deuxième point, c'est le manque de choix dans la banque d'états pouvant être collés qui fait en sorte que les images simulées présentent une faible variance. Nous remarquons que le choix de L_0 ne tient pas compte de la minimisation des erreurs d'agencement. En effet, nous considérons que les échelles $L_i < L_0$, $i = 1, 2, \ldots$, seront plus aptes à faire le raccordement fin.

Dans nos simulations, nous avons déterminé L_0 simplement par essais et erreurs pour ne garder que la plus petite grandeur L_0 permettant de reproduire la structure globale de l'image de référence.

6.2 Choix des recouvrements

Pour chacune des échelles utilisées, nous définissons un recouvrement Ω_i. De façon similaire aux recouvrements définis à la section 5.2.2, un recouvrement Ω_i est un ensemble de boîtes B_j qui couvre toute l'image simulée, excepté une bande de largeur $L_i/2$ aux bords de l'image. Plus précisément, un recouvrement Ω_i est l'union de deux sous-recouvrements $\Omega_{i,1}$ et $\Omega_{i,2}$ qui sont constitués de boîtes adjacentes (voir figure 6.1). La taille des boîtes à une échelle est la moitié de la taille à l'échelle précédente. Plus précisément, si L_0 est la plus

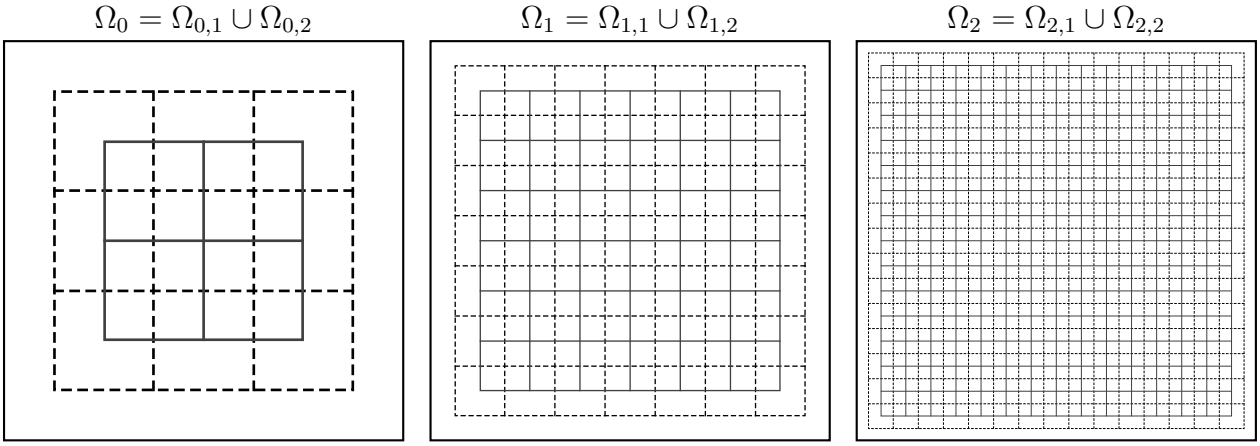


Figure 6.1 Les recouvrements utilisés pour une simulation multi-échelle. Les recouvrements en traits tiretés sont les $\Omega_{i,1}$ et les recouvrements en traits pleins sont les $\Omega_{i,2}$.

grande échelle, nous avons alors $L_i = {}^{L_0}/_{2^i}$, $i = 1, 2, \ldots$ Avec ce choix d'échelles et en laissant toujours une bande de taille ${}^{L_i}/_2$ autour des recouvrements, nous nous assurons que les recouvrements des différentes échelles restent alignés (voir figure 6.1). Cet alignement des recouvrements fait en sorte qu'une boîte du premier recouvrement de l'échelle $i+1$, c'est-à-dire une boîte $B_j \in \Omega_{i+1,1}$, est généralement positionnée au-dessus de l'intersection de quatre boîtes de l'échelle supérieure $B_k \in \Omega_i$ et permet donc de saisir les erreurs d'agencement observables aux frontières des états $\boldsymbol{s}$ collés à l'échelle i. De plus, une boîte du deuxième recouvrement de l'échelle $i+1$, c'est-à-dire une boîte $B_j \in \Omega_{i+1,2}$, correspond à un quartier de boîte de l'échelle supérieure $B_k \in \Omega_i$ et ne contient jamais de frontières entre les boîtes $B_k \in \Omega_i$. Ainsi, lorsque la simulation à l'échelle i est terminée, le recouvrement $\Omega_{i+1,2}$ ne contient que des quartiers d'états prélevés directement de l'image de référence et nous avons

$$e_{\mathrm{pc}}(j) = 0, \quad \forall\, j \in I_{\Omega_{i+1,2}}. \tag{6.1}$$

6.3 Conditionnement aux échelles supérieures

Afin d'éviter que les simulations aux échelles plus fines ne détruisent l'information structurelle laissée par les simulations aux échelles grossières, nous choisissons de fixer certaines

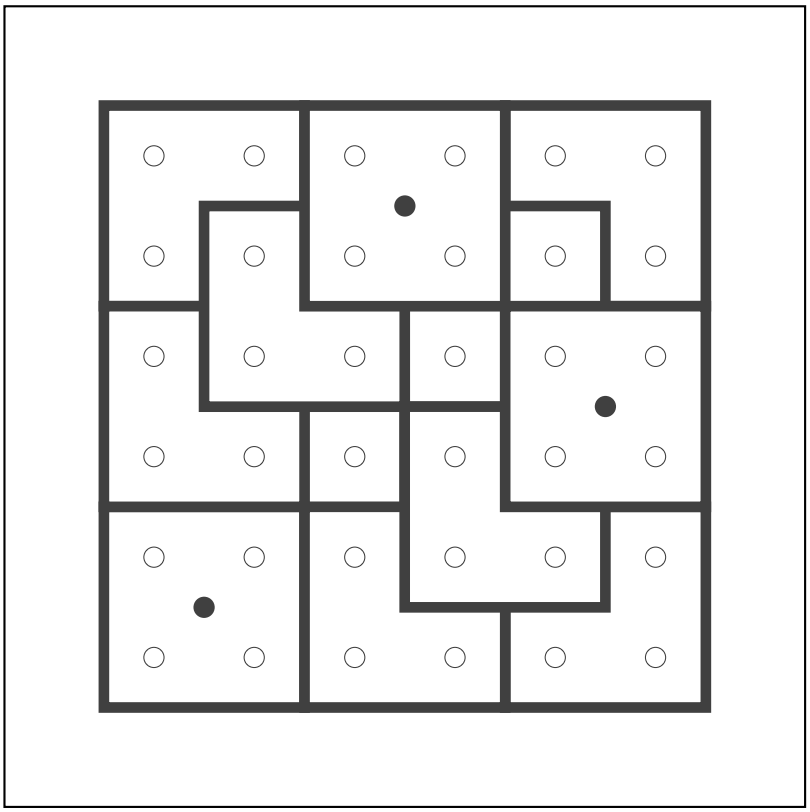


Figure 6.2 Illustration du collage résultant de la simulation de la plus grande échelle. Un disque blanc représente le centre d'une boîte $B_j \in \Omega_{1,2}$. Un disque noir représente le centre d'une boîte $B_j \in \Omega_{1,1}$ que nous sommes sûrs de fixer à la prochaine échelle.

données après chaque simulation d'échelle. Pour ce faire, les données situées dans une région présentant une petite erreur locale d'agencement sont fixées comme données conditionnantes. Plus précisément, après la simulation de l'échelle i, nous gelons les données centrales des boîtes $B_j \in \Omega_{i+1,1}$ telles que $e_{\mathrm{pc}}(j) = 0$. Puisque $e_{\mathrm{pc}}(j) = 0$ pour les boîtes $B_j \in \Omega_{i+1,2}$ (équation (6.1)), nous nous assurons que les données conditionnantes sont centrées dans une zone où les motifs sont continus dans un rayon d'au moins $L_i/2$. Par exemple, la figure 6.2 illustre par des disques blancs les centres des boîtes de $\Omega_{1,2}$ tout de suite après la simulation de l'échelle 0. Les cercles noirs représentent des emplacements (centrés dans les $B_j \in \Omega_{1,1}$) sûrs d'être fixés comme données conditionnantes.

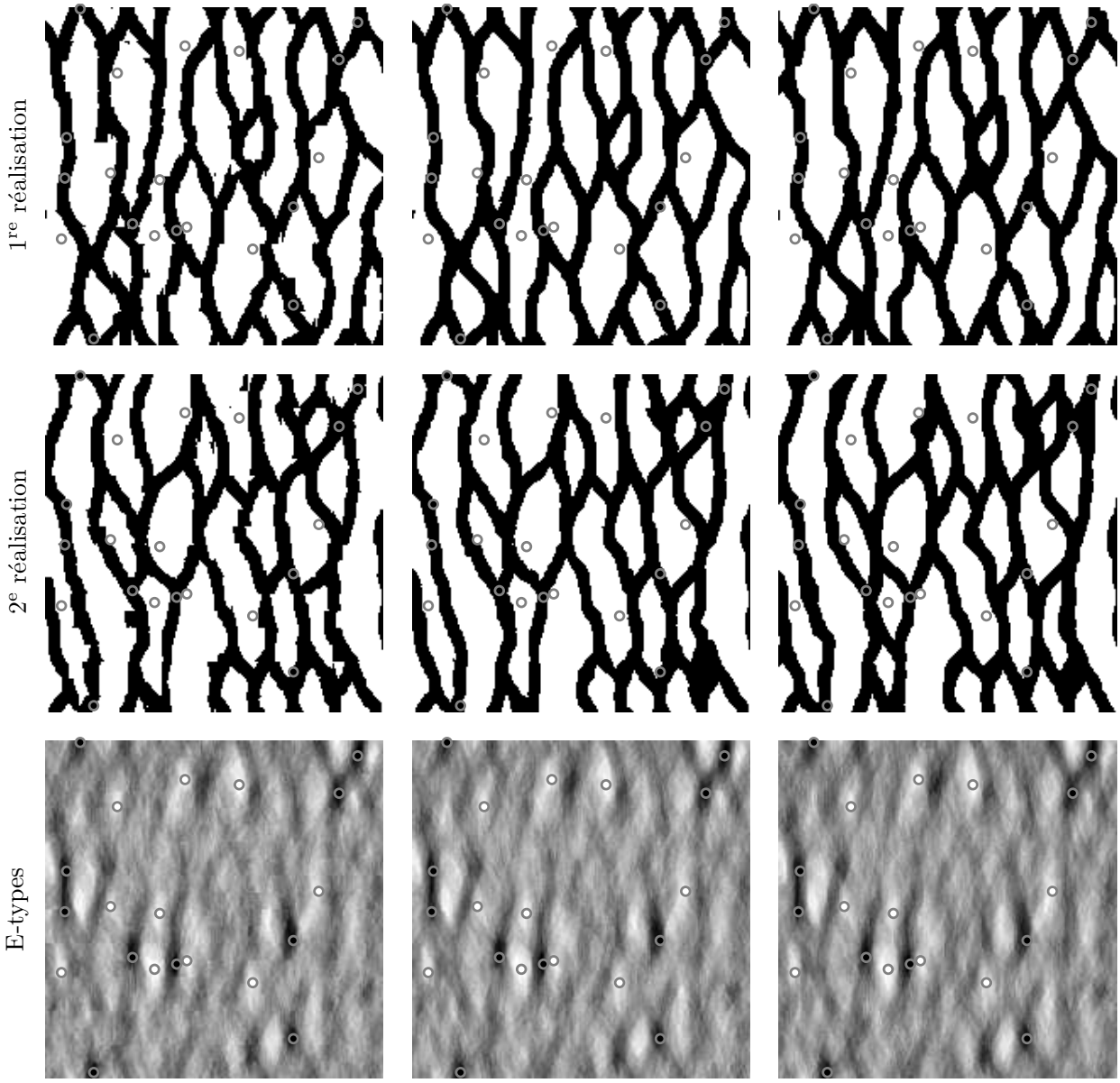


Figure 6.3 Simulations multi-échelles avec la méthode CPMS. Les données conditionnantes (hard data) sont représentées par des disques noirs ou blancs. Les deux premières lignes présentent de simples réalisations. La troisième ligne présente les images moyennes de 25 réalisations.

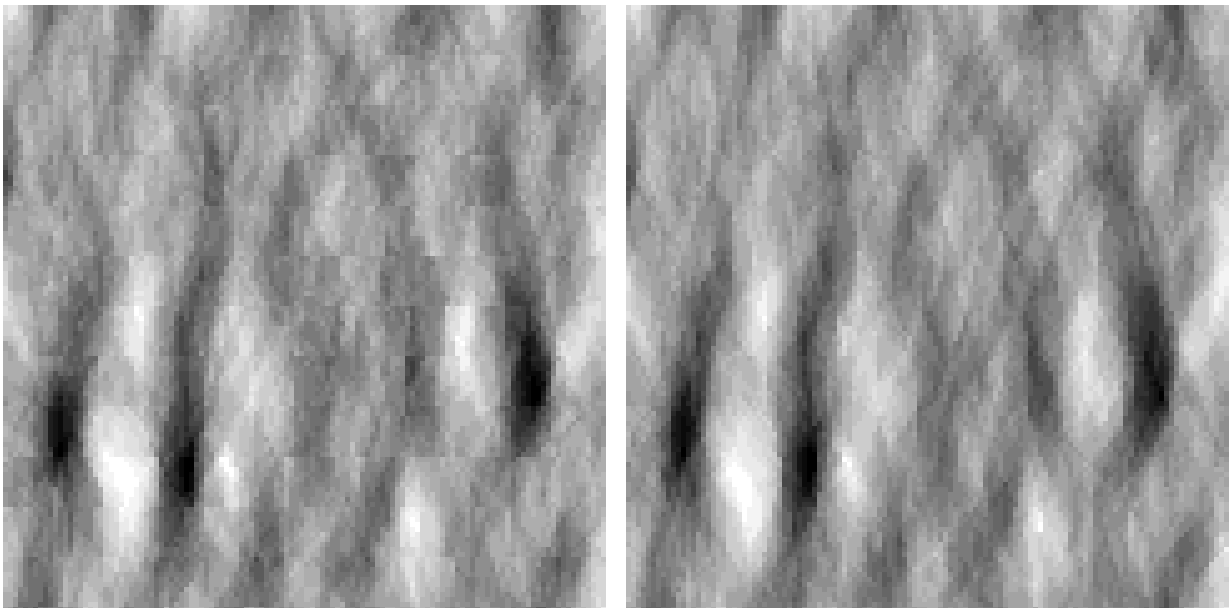
6.4 Simulations multi-échelles d'un réseau de chenaux

Nous présentons, à la figure 6.3, nos premiers résultats de simulations multi-échelles. Pour tester la méthode multi-échelle, nous utilisons la même image de référence qu'aux figures 5.13 et 5.14 de la section 5.8.5. Nous gardons $M = 2$ classes pour le contrôle de la moyenne locale et nous gardons $\boldsymbol{p}_t = \boldsymbol{p}_r$, c'est-à-dire que les images simulées doivent reproduire la même densité de chenaux que celle de l'image de référence. Pour ce test, nous n'utilisons que deux échelles : l'échelle 0 et l'échelle 1. Nous avons donc $M = 2$ et $\boldsymbol{p}_t = \boldsymbol{p}_r$ pour l'échelle 0 et l'échelle 1. Pour comparer avec nos résultats précédents, nous utilisons $L_0 = 20$ pixels et $L_1 = 10$ pixels, c'est-à-dire que nos deux échelles utilisent les mêmes tailles de boîtes que les deux premières séries de simulation de la figure 5.13.

La première et la deuxième ligne de la figure 6.3 présentent deux réalisations simples. La troisième ligne présente les moyennes de 25 réalisations. Cette image moyenne est appelée E-type. La première colonne présente les résultats de simulations après la simulation de la première échelle utilisant $L_0 = 20$. La deuxième et la troisième colonne présentent les résultats de simulations après le passage de la deuxième échelle $L_1 = 10$. Dans la deuxième colonne, les simulations à l'échelle 1 ne sont pas soumises au contrôle de l'histogramme, tandis que dans la troisième colonne, elles le sont.

La simulation de l'échelle 0 est toujours contrôlée et donne toujours une statistique $D^2 < 0.46$ où $0.46 = \chi^2_{1,50\%}$. Après simulation de l'échelle 1, nous réévaluons l'histogramme à l'échelle 0, c'est-à-dire avec des boîte de taille L_0. Si nous ne contrôlons pas la distribution à l'échelle 1 (colonne du milieu), nous observons une statistique médiane des 25 simulations à l'échelle 0 de $D^2 = 1.8$ (ne passe pas le test du χ^2). Par contre, si nous contrôlons la distribution à l'échelle 1 (colonne de droite), nous observons $D^2 = 0.1$ (ce qui est acceptable).

Les figures 6.3 et 6.4 montrent que les simulations multi-échelles permettent d'affiner efficacement le détail des structures de chenaux. À la figure 6.4a, où une seule échelle est utilisée, nous observons que l'image E-type présente des artéfacts carrés de taille 10×10 où $10 = {}^{L_0}/_2$. Ces carrés correspondent en fait à la grille utilisée (voir figure 6.1). À la figure 6.4b, nous observons que cet effet de grille semble avoir disparu après le passage de la deuxième échelle. Nous avons alors un résultat plus réaliste du milieu étudié.



(a) 1re échelle (b) 2e échelle

Figure 6.4 Détails de la partie centrale pour les images E-types. (a) Utilisation d'une seule échelle. (b) Utilisation de deux échelles.

CHAPITRE 7

DISCUSSION GÉNÉRALE

7.1 Probabilités de transition et choix de distances

Notre première méthode de simulation par rapiéçage de motif appelée *patchwork simulation method* (PSM) est une simulation unilatérale qui procède en collant des morceaux d'image de gauche à droite et de haut en bas. La méthode PSM repose sur un modèle de Markov où les probabilités de transition

$$p_{j,k} = \mathrm{P}\left[\boldsymbol{S}(n) = \boldsymbol{s}_k | \boldsymbol{U}(n) = \boldsymbol{u}_j\right]$$

définissent la probabilité de choisir l'état d'arrivée $\boldsymbol{S}(n)$ en fonction du voisinage immédiat : l'état de départ $\boldsymbol{U}(n)$. Ces probabilités ont été ajustées afin que les images produites par le processus soient compatibles avec l'histogramme visé défini par les probabilités $p_t(i)$. Cet histogramme visé de la moyenne locale peut être celui de l'image de référence ou tout autre histogramme défini par l'usager.

Quand le champ aléatoire étudié est homogène, c'est-à-dire que les différents états possibles du champ aléatoire peuvent être observés un nombre significatif de fois dans l'image de référence, nous pouvons considérer que le modèle probabiliste est pertinent. Dans ce cas, nous considérons que le vecteur aléatoire $\boldsymbol{S}_r$ [1] possède une distribution de probabilité représentative de la distribution du champ aléatoire étudié. En d'autres termes, pour un champ aléatoire homogène, nous considérons que

$$\boldsymbol{S}_r \overset{\mathrm{d}}{\simeq} \boldsymbol{Z}, \tag{7.1}$$

où $\boldsymbol{Z}$ est l'état d'un bloc $L \times L$ du champ aléatoire étudié.

Bien que le modèle probabiliste soit pertinent pour un champ homogène, il s'avère inutile

1. Rappelons que $\boldsymbol{S}_r$ est un état choisi au hasard dans l'image de référence de manière équiprobable.

pour la simulation de champs hétérogènes tels que le mélange de polymères (présentée aux sections 4.5.5 et 5.8.4). En effet, dans le cas de champs hétérogènes, les états $\boldsymbol{s} \in \mathcal{S}_r$ ne sont typiquement observés qu'une fois dans l'image de référence et nous ne pouvons plus supposer que $\boldsymbol{S}_r$ possède une distribution de probabilité représentative du champ aléatoire étudié. Une autre particularité des champs hétérogènes est que les états construits lors d'une simulation par rapiéçage ne sont généralement pas observables dans l'image de référence. Par exemple, l'état de départ $\boldsymbol{U}(n)$ dans la simulation PSM est le résultat d'un rapiéçage de cinq morceaux pigés dans l'image de référence (voir figure 4.5). Bien que cette construction pourrait être un état possible du champ aléatoire, il n'est généralement pas observable dans l'image de référence. Nous sommes donc généralement réduits à coller l'unique morceau qui s'agence le mieux avec le voisinage $\boldsymbol{U}(n)$, c'est-à-dire l'état $\boldsymbol{s} \in \mathcal{S}_r$ qui minimise $d(\boldsymbol{u}(\boldsymbol{s}), \boldsymbol{U}(n))$. Étant donné qu'un seul plus proche voisin est généralement trouvé, moduler les probabilités de choisir une classe plutôt qu'une autre ne permet pas d'influencer l'histogramme. Pour cette raison, la méthode de contrôle de l'histogramme appelée *adaptive-CHUSA* utilise des poids $\omega(i)$ afin de moduler les distances en fonction de la classe d'appartenance des états $\boldsymbol{s} \in \mathcal{S}_r$. Par exemple, si les états d'une classe d'indice i ont tendance à être surreprésentés, la méthode adaptive-CHUSA donne alors un poids $\omega(i)$ relativement faible par rapport aux poids des autres classes. Plus précisément, nous avons alors

$$\omega(i) < \omega(\ell), \ i \neq \ell,$$

de telle sorte que pour un état $\boldsymbol{s} \in C_i$ la distance modulée

$$\frac{d\left(\boldsymbol{u}(\boldsymbol{s}), \boldsymbol{u}_j\right)}{\omega(i)}$$

pénalise cette classe d'indice i (voir l'équation (4.29) de la section 4.4.4).

Cette idée de moduler les distances en fonction de la classe d'appartenance d'un état pour favoriser certaines classes est reprise dans notre méthode de simulation CPMS. Par contre, au lieu que la distance soit divisée par les poids $\omega(i)$, cette dernière est modulée en ajoutant un terme proportionnel à l'erreur relative $\epsilon(i)$. Rappelons que $\epsilon(i) = f(i) - p_t(i)$ est la différence entre $f(i)$ la fréquence des états appartenant à la $i^{\text{ième}}$ classe et $p_t(i)$ la probabilité visée pour

cette $i^{\text{ième}}$ classe. Plus précisément, dans la méthode CPMS, la distance utilisée pour un état $\boldsymbol{u} \in C_i$ est donnée par l'expression

$$d\,(\boldsymbol{u}, \boldsymbol{u}_j) \;+\; \gamma\,\epsilon(i),$$

où $\gamma\,\epsilon(i)$ est le terme de modulation (voir l'équation (5.17) de la section 5.4 pour les détails). Le contrôle de l'histogramme se fait donc uniquement par la modulation des distances. Si plus d'un état $\boldsymbol{u}$ minimise cette distance modulée, un de ces états est alors choisi au hasard et proportionnellement à son nombre d'occurrences dans l'image de référence. C'est le seul aspect que nous gardons de l'approche probabiliste.

7.2 Biais des simulations

Nous avons montré au chapitre 4 que la méthode unilatérale PSM introduit deux types de biais. Le premier est causé par la direction de la simulation, c'est-à-dire par un balayage allant de gauche à droite et de haut en bas. Ce balayage introduit une anisotropie dans les simulations qui peut être observée en particulier en présence de données conditionnantes. Dans ce cas, nous observons un phénomène d'ombre en bas à droite des données conditionnantes. Ce balayage directionnel fait aussi en sorte que la simulation unilatérale PSM se comporte différemment selon l'orientation des chenaux. Nous avons ainsi observé que la simulation unilatérale PSM se comporte mieux avec un réseau de chenaux orientés dans la direction sud-ouest nord-est, c'est-à-dire avec des chenaux orientés perpendiculairement à la direction de la simulation.

Un deuxième biais introduit par la simulation unilatérale PSM est causé par la grille de la simulation, c'est-à-dire par l'ensemble des emplacements espacés de $L/2$ où les motifs sont collés. Pour la simulation d'un réseau de chenaux, nous avons observé que les chenaux ont moins tendance à se situer à proximité de la frontière séparant les données existantes des données ajoutées. Cette tendance fait ressortir un effet de grille lors de l'observation de l'image moyenne d'un grand nombre de simulations (voir figures 4.19 et 5.15).

Pour remédier à ces lacunes de la méthode unilatérale, nous avons introduit la méthode itérative CPMS. Dans cette approche, la simulation ne suit plus un chemin prédéterminé,

mais visite plutôt les emplacements qui nécessitent des corrections. De ce fait, la simulation n'a pas de chemin orienté selon une direction précise et le biais observé dans la simulation unilatérale disparaît. Dans une simulation CPMS, l'algorithme débute typiquement par coller des motifs qui s'agencent avec les données conditionnantes. En plus de coller des morceaux au-dessus des données conditionnantes, la phase initiale d'une simulation CPMS consiste à nettoyer le bruit blanc de l'initialisation. Ce nettoyage crée un phénomène de grains, c'est-à-dire que l'image est un collage de petites régions localement bien assemblées (voir figure 5.6). La simulation tente ensuite de fusionner ces grains pour converger vers une image nette.

L'absence d'une vue d'ensemble empêche la méthode CPMS de bien raccorder les grains. En effet, la méthode étant une approche de type glouton, elle choisit un emplacement ayant une grande erreur locale $e(j)$ et ne tient pas compte des corrections apportées aux étapes précédentes. De plus, la petite correction locale effectuée à chaque étape ne tient pas compte des discontinuités qu'elle introduit dans son voisinage. Par exemple, dans le cas de la simulation du réseau de chenaux, l'algorithme peut se retrouver dans une boucle où il est constamment en train de construire et détruire un chenal sans jamais réussir à le raccorder au reste du réseau. C'est aussi la raison pour laquelle les grains de l'échiquier de la figure 5.6 se stabilisent à une taille donnée.

7.3 Temps de calcul et espace mémoire

Un élément important développé dans nos méthodes de simulation est l'implémentation d'arbres de recherche. Ces arbres sont utilisés afin de classer les morceaux de l'image de référence pour chercher rapidement le morceau le plus proche du morceau observé dans l'image simulée. Cette structure arborescente est cruciale pour effectuer un grand nombre de simulations dans un laps de temps raisonnable (environ 1 s pour une simulation unilatérale et 20 s pour une simulation itérative [2]). Par exemple, pour la simulation unilatérale, nous pouvions aisément faire rouler 256 simulations d'images 200×200 et observer la moyenne.

Cependant, le prix à payer pour accélérer le temps de calcul est l'espace mémoire requis pour stocker la structure des arbres de recherche. C'est d'ailleurs la raison pour laquelle nos

2. Nos simulations ont été effectuées sur un MacBook Air avec un processeur Intel Core i5 à 1.7 GHz et 4 Go de mémoire. Le code a été implémenté en C++.

simulations sont limitées à des images de dimension 200×200. En effet, pour une image de référence de dimension 200×200, plus de 270 000 morceaux d'image 16×16 peuvent être observés (en considérant les huit symétries). Enregistrer ces états avec la structure de nos arbres de recherche peut nécessiter entre 500 Mo et 1 Go de mémoire vive.

Pour nos simulations, nous avons utilisé deux structures d'arbre de recherche décrites par Liu, Moore, Gray et Yang (2004) appelées arbres métriques et arbres étendus. Dans un arbre métrique, nous effectuons une recherche classique en profondeur d'abord (*depth-first search*). Ce type d'arbre donne une bonne accélération à la recherche des plus proches voisins, mais n'est pas très efficace pour des données en grande dimension (comme nos morceaux d'image composés de plus de 64 pixels). L'arbre étendu (*spill tree*) prend plus d'espace mémoire, mais évite les visites dans les branches voisines de l'arbre. En se concentrant sur la visite d'un nombre limité de branches, l'arbre étendu permet une recherche plus rapide que la recherche avec un arbre métrique. Nous avons observé que l'arbre étendu est particulièrement efficace pour les champs homogènes tels que nos simulations de disques ou nos simulations de réseaux de chenaux (environ $10\times$ plus rapide). Cependant, comme l'espace mémoire requis devient important (plus de 2 Go) pour la simulation d'un champ hétérogène tel que le mélange de polymères, nous avons alors utilisé des arbres métriques. Il faut donc faire un compromis entre temps de calcul et espace mémoire utilisé.

Dans ce contexte, les simulations multi-échelles présentées au chapitre 6 étaient les plus grosses simulations que nous pouvions faire, car elles ont utilisé toute la mémoire disponible.

CONCLUSION

Dans cette thèse, nous avons présenté deux méthodes de simulations par rapiéçage de motifs. Ces méthodes, à la manière d'un casse-tête, construisent des images en collant des morceaux carrés d'image prélevés dans une image de référence.

Notre première approche appelée *patchwork simulation method* (PSM) procède de manière unilatérale en collant des morceaux d'image de gauche à droite et de haut en bas. Cette première méthode repose sur un modèle de Markov où les probabilités de transition dépendent du voisinage immédiat. Ces dernières probabilités ont été ajustées afin que les images produites par le processus respectent les statistiques désirées. Plus précisément, les images peuvent reproduire un histogramme de la moyenne locale défini par l'usager.

Notre deuxième approche appelée *corrective pattern-matching simulation* (CPMS) procède de manière itérative en effectuant des corrections aux emplacements où une grande erreur locale a été mesurée. Cette deuxième méthode repose sur un algorithme de type glouton. En effet, à chaque étape de la simulation, nous faisons un choix local optimal dans l'espérance de tendre vers un optimum global. L'erreur locale considérée tient compte de trois facteurs : la conformité aux données conditionnantes (*hard data*) ; la conformité aux motifs retrouvés dans l'image de référence et la conformité à une mesure globale, l'histogramme de la moyenne locale.

Nos méthodes de simulation présentent diverses améliorations notables par rapport aux méthodes de simulation multi-point existantes.

Premièrement, nos méthodes mettent l'accent sur la continuité des motifs produits dans l'image simulée. Cette continuité est assurée en donnant plus de poids au pourtour de la région de collage. Nous avons observé que nos simulations donnent des résultats visuellement fidèles aux motifs observés dans l'image de référence.

Une deuxième amélioration est l'adaptation de la méthode unilatérale PSM aux données conditionnantes. En considérant les données conditionnantes se trouvant dans une région étendue (*extended block-state*), la méthode unilatérale peut choisir le morceau à coller en fonction des données futures.

Une troisième amélioration apportée par nos méthodes de simulation est notre méthode de contrôle de l'histogramme de la moyenne locale. Dans la méthode unilatérale PSM, le contrôle est effectué en ajustant convenablement les probabilités de transition. Dans la méthode itérative CPMS, le contrôle est assuré par l'erreur locale statistique. Cette erreur locale statistique, en étant pondérée adéquatement, donne juste assez d'importance au contrôle de l'histogramme pour ne pas entraver la minimisation de l'erreur sur la continuité des motifs (*pattern conformity error*). Par la succession de corrections locales apportées, l'algorithme CPMS fait tendre l'image simulée vers une image ayant un histogramme fidèle à l'histogramme voulu. Les méthodes de contrôle de l'histogramme que nous avons proposées sont flexibles, car elles permettent à l'usager de choisir une distribution voulue et créer ainsi des images qui diffèrent significativement de l'image de référence. Dans le cas particulier où l'objectif est de reproduire l'histogramme de l'image de référence, nos images simulées sont fidèles à l'aspect visuel de l'image de référence.

Un quatrième point important développé dans nos méthodes de simulation est l'implémentation d'arbres de recherche. Ces arbres sont utilisés afin de classer les morceaux de l'image de référence. Grâce à ces arbres, nous pouvons trouver rapidement le morceau s'agençant le mieux avec un autre. Nous avons donc pu effectuer un grand nombre de simulations dans un laps de temps raisonnable.

Finalement, en appliquant l'algorithme CPMS à deux échelles, nous avons pu produire des réseaux de chenaux lisses tout en conservant un histogramme de la moyenne locale statistiquement acceptable. Notre approche multi-échelle débute par une simulation avec une taille de motif suffisamment grande pour reproduire la structure générale de l'image de référence. Ensuite, certains points de l'image simulée sont figés pour être utilisés comme données conditionnantes pour l'échelle suivante. En réappliquant la méthode CPMS avec des motifs de plus petite taille, nous pouvons affiner les détails de l'image simulée.

Améliorations futures

Bien que notre méthode de simulation CPMS nous ait permis de réaliser des simulations respectant un bon nombre de propriétés statistiques, nous avons constaté que cette méthode comporte une lacune importante : son absence de mémoire. Rien ne guide le chemin suivi pour

apporter les corrections, si ce n'est l'erreur locale. De manière similaire à un algorithme tabou où l'on garde la trace des modifications précédentes, nous pourrions améliorer la méthode en donnant une certaine mémoire à l'algorithme. Ceci permettrait de compléter de proche en proche des régions de l'image simulée et ainsi éviter d'osciller dans un minimum local. Compléter l'image de proche en proche est d'ailleurs un point fort de la méthode unilatérale PSM. En allant dans une direction précise, cette dernière pourrait construire de grandes structures en rapiéçant bout à bout des morceaux d'image.

Une autre lacune avec nos méthodes de simulation est le respect des données conditionnantes. Dans la plupart des résultats présentés, nous avons considéré que les données conditionnantes étaient compatibles avec les motifs observés dans l'image de référence. Cependant, lorsqu'il est impossible de trouver un morceau de l'image de référence qui est compatible avec des données conditionnantes, nos méthodes de simulation prennent simplement un des morceaux qui respectent le plus grand nombre de données conditionnantes. Les réalisations présentent alors des données conditionnantes non respectées (telles qu'à la figure 4.19).

Pour rendre nos méthodes de simulation plus robustes, nous devrions trouver le moyen d'adapter la simulation à des données conditionnantes quelconques. Nous pourrions adopter une approche semblable à notre approche multi-échelle. Si aucun morceau d'image de taille $L \times L$ ne peut s'agencer avec les données conditionnantes, nous pourrions alors essayer de coller de plus petits morceaux $L/2 \times L/2$. En effet, plus les morceaux sont petits, plus il est facile de satisfaire localement les données conditionnantes.

Afin de satisfaire des données conditionnantes incompatibles avec l'image de référence, nous pourrions de plus enrichir la banque d'états disponibles à l'aide de simulations non conditionnelles. Ces simulations non conditionnelles pourraient générer de nouveaux états qui seraient compatibles avec les données conditionnantes.

Une autre amélioration facilement envisageable pour nos méthodes de simulation serait de considérer plus de deux faciès, c'est-à-dire plus de deux valeurs possibles pour les pixels. La principale modification avec l'ajout de faciès, c'est la métrique utilisée pour la distance. Il faudrait alors définir les distances $d_{a,b}$ relatives aux faciès a et b. Cependant, en ajoutant des faciès, l'entropie de l'image augmenterait et une image de référence de plus grande taille serait nécessaire pour obtenir des simulations de qualité semblable.

Finalement, une amélioration à apporter à nos méthodes de simulation serait de transposer leur implémentation en trois dimensions. Bien que nos méthodes de simulations ne consistent qu'à produire des champs aléatoires en deux dimensions, elles pourraient aisément se transposer en trois dimensions. Ainsi, au lieu de coller des carreaux, nous collerions des cubes de taille $L \times L \times L$ dans une image 3D. Les poids utilisés pour considérer le voisinage seraient concentrés autour de la surface du cube de côté L. Le problème avec une extension 3D de nos méthodes serait l'espace mémoire requis pour stocker les arbres de recherche. En effet, si l'on considérait des volumes d'au moins $100 \times 100 \times 100$, il faudrait plus de 16 Go d'espace mémoire pour enregistrer tous les états. Une façon de limiter l'espace mémoire utilisé serait d'envisager une approche semblable à la méthode d'échantillonnage directe (*direct sampling method*). En effet, en utilisant cette méthode, au lieu de créer des structures d'arbre de recherche, le volume de référence serait balayé à partir d'emplacements choisis au hasard d'où nous pourrions trouver des données compatibles.

RÉFÉRENCES

ARPAT, G. et CAERS, J. (2005). *Sequential simulation with patterns*. Stanford University.

ARPAT, G. et CAERS, J. (2007). Conditional simulation with patterns. *Mathematical Geology*, 39, 177–203.

BOUCHER, A. et STRIGHT, L. (2008). Downscaling with training image and search tree partitioning. *Proceeding of Geostats 2008*. Santiago, Chile.

CHILÈS, J.-P. et DELFINER, P. (1999). *Geostatistics : modeling spatial uncertainty*. Wiley-Interscience.

CHUGUNOVA, T. (2008). *Contraintes des modèles génétiques de réservoirs par une approche de reconnaissance statistique de forme*. École des Mines de Paris.

DEUTSCH, C. et JOURNEL, A. (1998). *GSLIB : Geostatistical Software Library and User's Guide*. Oxford University Press, seconde édition.

EL OUASSINI, A. (2006). *Une nouvelle approche pour la reconstruction statistique des images : le rapiéçage de motifs stochastiques*. Mémoire de maîtrise, École Polytechnique de Montréal.

EL OUASSINI, A., SAUCIER, A., MARCOTTE, D. et FAVIS, B. (2008). A patchwork approach to stochastic simulation : A route towards the analysis of morphology in multiphase systems. *Chaos, Solitons & Fractals*, 36, 418–436.

EMERY, X. (2004). Properties and limitations of sequential indicator simulation. *Stochastic Environmental Research and Risk Assessment*, 18, 414–424.

FAUCHER, C., SAUCIER, A. et MARCOTTE, D. (2012). A new patchwork simulation method with control of the local-mean histogram. *Stochastic Environmental Research and Risk Assessment*. 10.1007/s00477-012-0586-9.

GUARDIANO, F. et SRIVASTAVA, R. (1993). Multivariate geostatistics : Beyond bivariate moments. *Geostatistics-Troia*, 1, 133–144. In Soares.

JOURNEL, A. (1993). Geostatistics : Roadblocks and challenges. *Geostatistics-Troia*, 1, 212–224. In Soares.

LIU, T., MOORE, A., GRAY, A. et YANG, K. (2004). An investigation of practical approximate nearest neighbor algorithms. *Proceedings of NIPS 2004*. Vancouver.

MARIETHOZ, G., RENARD, P. et STRAUBHAAR, J. (2010). The direct sampling method to perform multiple-point geostatistical simulations. *Water Resour. Res.*, 46.

PARRA, A. et ORTIZ, J. (2011). Adapting a texture synthesis algorithm for conditional multiple point geostatistical simulation. *Stochastic Environmental Research and Risk Assessment*, 25, 1101–1111. 10.1007/s00477-011-0489-1.

PICKARD, D. (1980). Unilateral Markov fields. *Advances in Applied Probability*, 12, 655–671.

PRESS, W., TEUKOLSKY, S., VETTERLING, W. et FLANNERY, B. (2007). *Numerical recipes : the art of scientific computing*. Cambridge Univ Press, troisième édition.

RAZLIGHI, Q., KEHTARNAVAZ, N. et NOSRATINIA, A. (2009). Computation of image spatial entropy using quadrilateral Markov random field. *IEEE Transactions on Image Processing*, 18, 2629–2639.

SAITO, H., MCKENNA, S., ZIMMERMAN, D. et COBURN, T. (2005). Geostatistical interpolation of object counts collected from multiple strip transects : Ordinary kriging versus finite domain kriging. *Stochastic Environmental Research and Risk Assessment*, 19, 71–85.

SAUCIER, A., RICHER, J. et MULLER, J. (2002). Assesing the scope of the multifractal approach to textural characterization with statistical reconstructions of images. *Physica A : Statistical Mechanics and its Applications*, 311, 231 – 259.

STREBELLE, S. (2000). *Sequential Simulation Drawing Structures from Training images*. Thèse de doctorat, Stanford University.

STREBELLE, S. (2002). Conditional simulation of complex geological structures using multiple-point statistics. *Mathematical Geology*, 34, 1–21.

STREBELLE, S. (2003). New multiple-point statistics simulation implementation to reduce memory and cpu-time demand. *Conference of the international association for mathematical geology. Portsmouth, UK*. vol. 7, 12.

TAHMASEBI, P., HEZARKHANI, A. et SAHIMI, M. (2012). Multiple-point geostatistical modeling based on the cross-correlation functions. *Computational Geosciences*, 16, 779–797.

TUANFENG, Z., STEIN, I. et MCCORMICK, D. (2008). Patched path and recursive servo system in multi-point geostatistical simulation. *Proceeding of Geostats 2008*. Santiago, Chile.

WEI, L. et LEVOY, M. (2000). Fast texture synthesis using tree-structured vector quantization. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 479–488.

WU, J., BOUCHER, A. et ZHANG, T. (2008). A sgems code for pattern simulation of continuous and categorical variables : Filtersim. *Computers & Geosciences*, 34, 1863–1876.

ZHANG, C. et LI, W. (2008). A comparative study of nonlinear markov chain models for conditional simulation of multinomial classes from regular samples. *Stochastic Environmental Research and Risk Assessment*, 22, 217–230.

ZHANG, T., SWITZER, P. et JOURNEL, A. (2006). Filter-based classification of training image patterns for spatial simulation. *Mathematical Geology*, 18, 63–80.